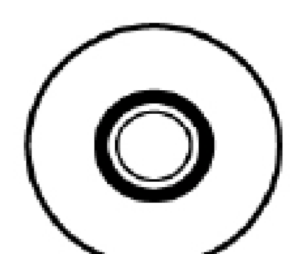
原创经典

以全新视觉展示P2P技术从入门到应用实践的学习之路
全面揭秘NAT、BT、eMule、Skype、流媒体5大关键技术
配合实际开发案例，引导读者进行P2P应用实战开发

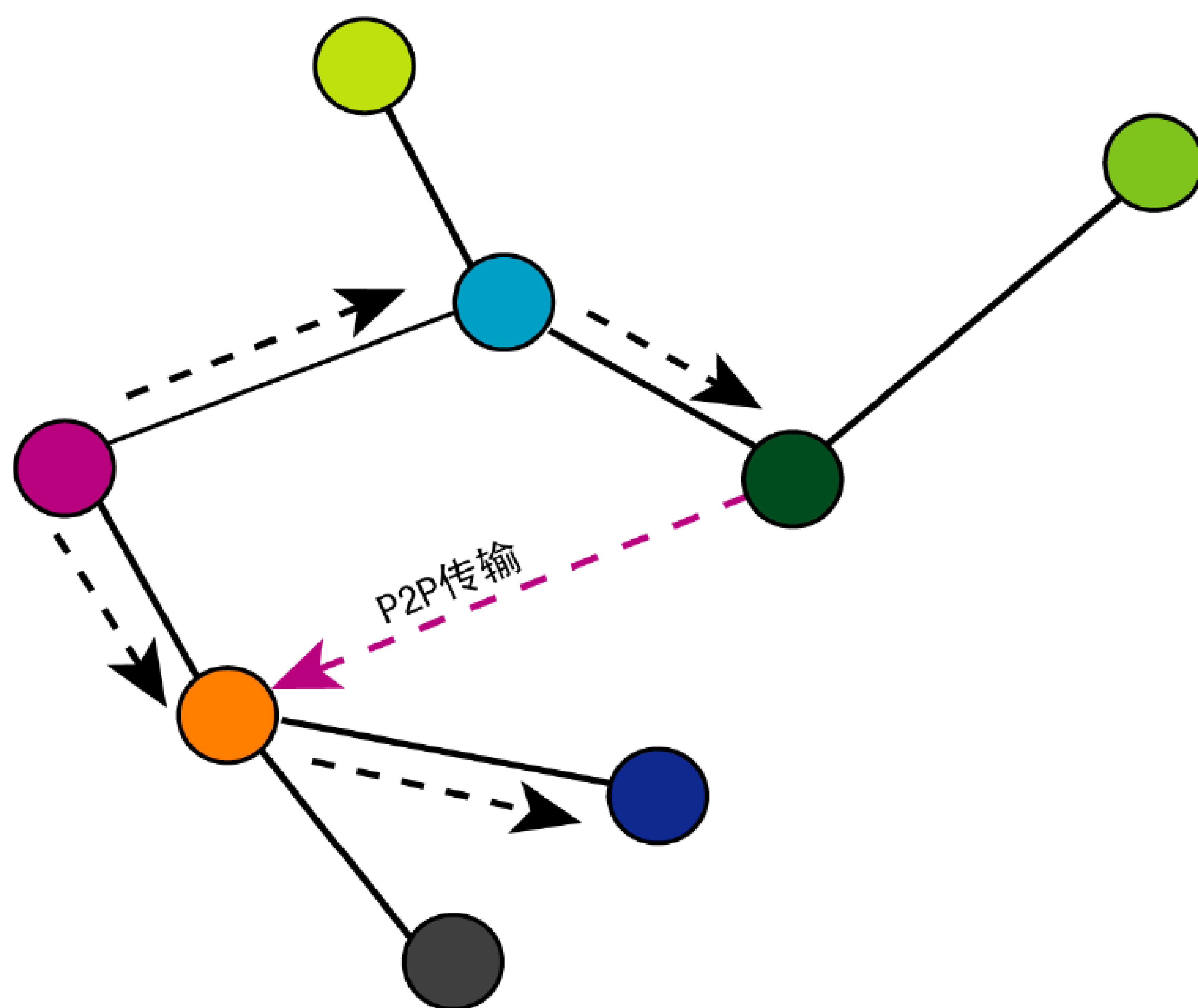
P2P技术揭秘

——P2P网络技术原理与典型系统开发



（11.5小时多媒体语音教学视频）

管 磊　等编著



清华大学出版社

P2P 技术揭秘

——P2P 网络技术原理与典型系统开发

管磊　编著

清华大学出版社
北　京

内 容 简 介

本书从一个全新的视角向读者展示了从 P2P 初步入门到应用实践的学习之路。本书从理论到实践，从基础到项目，循序渐进地讲解了 P2P 技术的基本知识体系，同时配合开发案例引导读者进行 P2P 应用实战开发。书中对每一个知识点、原理思想、应用方法及实例都进行了深入浅出的阐述和分析，力求让读者读完本书后有所学、有所悟、有所得。

本书共 14 章，分为 3 篇。主要内容包括 P2P 概述、P2P 网络拓扑结构、P2P 网络搜索技术、P2P 关键技术及应用、P2P 网络中的 NAT 穿透技术、基于 P2P 的 BitTorrent（后文简称 BT）技术、基于 P2P 的 eMule 文件共享技术、基于 P2P 的 Skype 即时通信技术、基于 P2P 的流媒体技术、基于 Java 的 P2P 开发平台搭建、Skype 的开发包及插件开发技术、基于 P2P 的即时通信系统的开发与实现、BT 系统分析及客户端开发、JXTA 技术等。另外，本书配书光盘中收录了专门为本书录制的多媒体教学视频及书中涉及的源代码，便于读者更加直观、高效地学习。

本书适合 P2P 技术入门人员及网络视频、网络电话、多线程下载等网络软件开发人员。另外，本书对于大中专院校相关专业的学生和老师也有很好的借鉴意义。

图书在版编目（CIP）数据

P2P 技术揭秘——P2P 网络技术原理与典型系统开发 / 管磊编著. —北京：清华大学出版社，2010.10
ISBN 978-7-302-22830-1

Ⅰ. ①P…　Ⅱ. ①管…　Ⅲ. ①因特网－基本知识　Ⅳ. ①TP393.4

中国版本图书馆 CIP 数据核字（2010）第 097099 号

责任编辑：夏兆彦
责任校对：徐俊伟
责任印制：
出版发行：清华大学出版社　　**地　址：**北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**邮　编：**100084
社 总 机：010-62770175　　**邮　购：**010-62786544
投稿与读者服务：010-62795954，jsjjc@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：
装 订 者：
经　销：全国新华书店
开　本：185×260　**印　张：**41　**字　数：**1024 千字
（附光盘 1 张）
版　次：2010 年 10 月第 1 版　**印　次：**2010 年 10 月第 1 次印刷
印　数：
定　价：　元

产品编号：036233-01

前　　言

当你正在用 QQ 尽情地聊天时，当你用酷狗欣赏着美妙的音乐时，当你用迅雷神速地下载一部高清电影时……你是否意识到你正在享受着 P2P 带来的快感与兴奋?

为什么要写作这本书?

对等网络（P2P）被美国《财富》杂志称为改变因特网发展的四大新技术之一。P2P 技术不仅为人们提供了前所未有的自由和便利，同时也有效地整合了互联网的潜在资源，将基于网页的互联网转变成动态存取、自由交互的海量信息网络。及早关注、跟踪、学习、研究这一技术的发展，才有可能较为从容地面对它所带来的冲击。同时，伴随着 P2P 技术在商业应用中的发展，它所展示出来的巨大商业潜力和价值也是不可估量的。掌握了 P2P 技术，也就掌握了这一领域的致胜之道和创造财富之道。

P2P 技术方兴未艾，各种基于 P2P 技术的应用风起云涌。在当前网络发展的大潮中，P2P 技术以其优异的特性和一种对等共享的思想在互联网发展中扮演了非常重要的角色。真正的 P2P 应用程序能够让具有创新意识的小团队开发出能与大公司相抗衡的软件和业务；真正的 P2P 技术应用于成熟市场后，将会是一种颠覆性的技术，其能量不可估量。

目前专门论述 P2P 技术原理和实践的图书非常少。本书便是基于这种背景而编写的。希望本书能成为广大迫切希望学习和研究 P2P 技术和应用的读者的良师益友，将一个全面、详尽、清晰、透彻的 P2P 技术完整地展现给读者，并指导读者进行 P2P 应用实践。

学习 P2P 技术，不仅要学习它的基本理论与技术，更要理解它的原理，掌握它的应用。本书从 P2P 的基础理论、技术应用和实践开发三个方面进行讲解，力求通俗易懂，深入浅出，把道理讲明白，把技术说透彻，把实践描述完整，给读者展示一个清晰的 P2P 技术世界。本书将 P2P 的理论与实际应用紧密结合起来，透析 P2P 的原理和机制，让 BT、eMule、Skype 等看似神秘的东西不再神秘，让你也可以近距离地看清它的本质，甚至可以自己去实现。另外，本书的实践环节提供了大量实用、典型的热门应用案例引导读学习和实践，提高 P2P 的应用开发水平。希望在本书的指引下，让你拓展视野，思考 P2P 对互联网的影响，发现 P2P 的力量和潜在的价值。

本书有何特色?

本书重点突出，主次分明，对学习过程中容易出现的疑点和难点进行了详细说明。选择的实例和案例力求以解决实际问题为导向，以项目实践为目的，对读者理解 P2P 技术和

应用开发有很大的帮助。全书讲解方式简单直接，从基础理论到基本应用，再到项目实践，由浅入深，循序渐进，力求让读者全面掌握 P2P 技术的整个知识体系。下面具体介绍一下本书的主要特色。

1. 配大量多媒体教学视频，高效、直观

作者专门为本书录制了大量多媒体教学视频，便于读者高效、直观地学习。这些视频连同本书涉及的源代码一起收录于本书的配书光盘中。

2. 内容全面、系统，知识要点丰富

本书不是针对 P2P 的某一技术点，而是对 P2P 技术要素、基本原理、应用等方面进行全面、系统的讲解，力求把 P2P 的整个技术体系展现给读者。例如，讲解 P2P 技术基础与原理的同时，还讲解了 NAT、流媒体、即时通信等技术，并对 P2P 应用开发中涉及的 Java、Linux 和 Eclipse 等技术也进行了必要讲解，便于读者系统地理解 P2P 的技术体系和应用开发。

3. 难易适中、读者定位准确

本书主要针对 P2P 初学者，写作时从入门者的角度重点讲解 P2P 的基础知识体系，让读者在应用的基础上理解内在的技术原理。而对于一些复杂的深度理论本书不涉及或者较少涉及。

4. 理论结合实践，重在应用

本书不是一味地讲技术，也不是一味地讲实践。而是理论结合实践，将理论知识及技术原理渗透到应用案例中，便于读者更好地理解 P2P 应用开发。

5. 与市场需求相结合，实用价值高

本书所讲解的应用技术都是当前应用广泛、非常热门的技术，如 NAT、BT、eMule、Skype、流媒体等。它们不仅是一种技术，更是一种盈利的商业平台，有很高的实用价值，拥有巨大的市场潜力。学习了这些技术，对工作、学习及就业都将有很高的价值。

6. 案例丰富、典型

提供了 Skype 开发包及插件开发、即时通信系统开发、BT 系统分析及客户端开发三个典型案例，便于读者深入理解基于 P2P 技术开发的基本原理、基本方法、编程技巧及完整的实现过程。

本书内容及知识体系

第 1 篇　基础理论篇（第 1～4 章）

本篇主要介绍了 P2P 的基本概念和技术等基础理论。主要内容包括：P2P 概述、P2P

的网络拓扑结构、P2P 网络的搜索技术、P2P 的关键技术及应用。

第 2 篇　技术应用篇（第 5～9 章）

本篇内容是 P2P 技术的精髓所在，主要介绍了 P2P 技术在互联网和生活中的典型应用。主要内容包括：P2P 网络中的 NAT 穿透技术、基于 P2P 的 BT 技术解析、基于 P2P 的 eMule 文件共享技术、基于 P2P 的 Skype 即时通信技术、基于 P2P 的流媒体技术。

第 3 篇　实践开发篇（第 10～14 章）

本篇以典型案例的形式讲解 P2P 应用系统的开发方法，真正解决如何进行 P2P 开发的问题。主要内容包括：基于 Java 的 P2P 开发平台搭建、Skype 的开发包及插件开发技术、基于 P2P 的即时通信系统的开发与实现、BT 系统分析及客户端开发、JXTA 技术概述。

配书光盘内容介绍

为了便于读者使用本书，本书附带 1 张 DVD 光盘。内容如下：

- 11.5 小时配套多媒体语音教学视频；
- 本书所涉及的实例和项目案例的源代码；
- 本书实例与项目案例开发及运行所需的开发包。

本书读者对象

- 没有任何基础的 P2P 技术初学者；
- 有一定基础，想进一步系统学习 P2P 技术的人员；
- 从事 P2P 应用开发的人员；
- 网络视频、网络电话、多线程下载等网络软件开发人员；
- 大中专院校相关专业的学生和老师；
- 相关培训班的学员。

本书阅读建议

- 没有网络基础的读者，先学习一下计算机网络方面的基础知识，了解一下网络的基础结构后再学习本书内容；
- 入门读者建议从本书第 1 章开始按章节顺次阅读，效果更好；
- 有 P2P 技术基础的读者，可以根据实际情况有重点地选择阅读；
- 对于书中的实例和案例，读者可以先自己思考一下如何实现，然后再阅读相关内容，并结合本书多媒体教学视频和源代码，亲自动手实现一次，理解会更加深刻。

本书作者及编委会成员

本书由管磊主笔编写。其他参与编写和资料整理的人员有武冬、郅晓娜、孙美芹、卫丽行、尹翠翠、蔡继文、陈晓宇、迟剑、邓薇、郭利魁、金贞姬、李敬才、李萍、刘敬、陈慧、刘艳飞、吕博、全哲、佘勇、宋学江、王浩、王康、王楠、杨宗芳、张严虎、周玉、张平、张靖波、周芳、杨罡、于海滨、张晶杰、张利峰、杨景凤、陈锴、郑剑锋、叶佩思、张涛、赵东彪、王双。在此一并表示感谢！

本书编委会成员有欧振旭、陈杰、陈冠军、项宇峰、张帆、陈刚、程彩红、毛红娟、聂庆亮、王志娟、武文娟、颜盟盟、姚志娟、尹继平、张昆、张薛。

编著者

目　　录

第 1 篇　基础理论篇

第 2 篇 技术应用篇

第 3 篇 实战开发篇

第 1 篇　基础理论篇

第 1 章　走进 P2P 的世界

计算机对等网络技术（P2P（Peer-To-Peer）技术）是目前计算机网络技术研究领域的热点，曾经被《财富》杂志誉为将改变因特网未来的四大新技术之一。作为一种新兴技术，P2P 技术因能充分利用网络资源及对等利用、资源共享等诸多优点而受到广泛关注。本章将系统地讲解 P2P 的基础知识，带领读者敲开网络王国的大门，走进神奇的 P2P 网络世界。本章主要讲解的基础知识如下。

- ❑ Internet 的起源与发展：了解网络的基础知识与 Internet 的起源。
- ❑ P2P 的概念理解：从 Internet 的起源引出 P2P 的概念，理解 P2P 的思想内涵。
- ❑ P2P 网络与传统网络对比：要理解 P2P 网络与普通的 Web 网络的异同。
- ❑ P2P 的发展历程：通过 P2P 技术的发展与演进，了解 P2P 在不同历史时期的发展特点，了解当前互联网上常见的 P2P 应用。
- ❑ P2P 网络分类及特点：掌握 P2P 网络的分类标准，理解不同类别的网络结构特点，理解 P2P 的技术特点。
- ❑ P2P 的未来发展：了解 P2P 技术的未来发展、应用前景及研发情况等。

1.1　从 Internet 说起

要介绍一个新事务，必须首先回答一个问题，它是从何而来的，本节就从 Internet 说起，从而引出 P2P 网络的历史渊源。

1.1.1　Internet 的起源

因特网是 Internet 的中文译名，它的前身是美国国防部高级研究计划局（ARPA）主持研制的 ARPAnet。“阿帕网”（ARPAnet）于 1969 年正式启用，当时仅连接了 4 台计算机，供科学家们进行计算机联网实验使用。这就是因特网的前身。

从 ARPAnet 发展到今天的因特网，可以用图 1.1 简要描述 Internet 的发展流程。

图 1.1 描述了 Internet 从起源到发展的大致过程，需要指出的是，在这个过程中，尽管后来的 NSFnet 取代了 ARPAnet 成为 Internet 的核心，但 ARPAnet 中所使用的 TCP/IP 技术被证明是非常成功的，并且一直沿用至今。

在 1992 年的时候，美国的 IBM、MCI、MERIT 三家公司联合组建了一个高级网络服务公司（ANS），建立了一个新的网络，叫做 ANSnet，成为 Internet 的另一个主干网，这标志着 Internet 正式走向商业化。

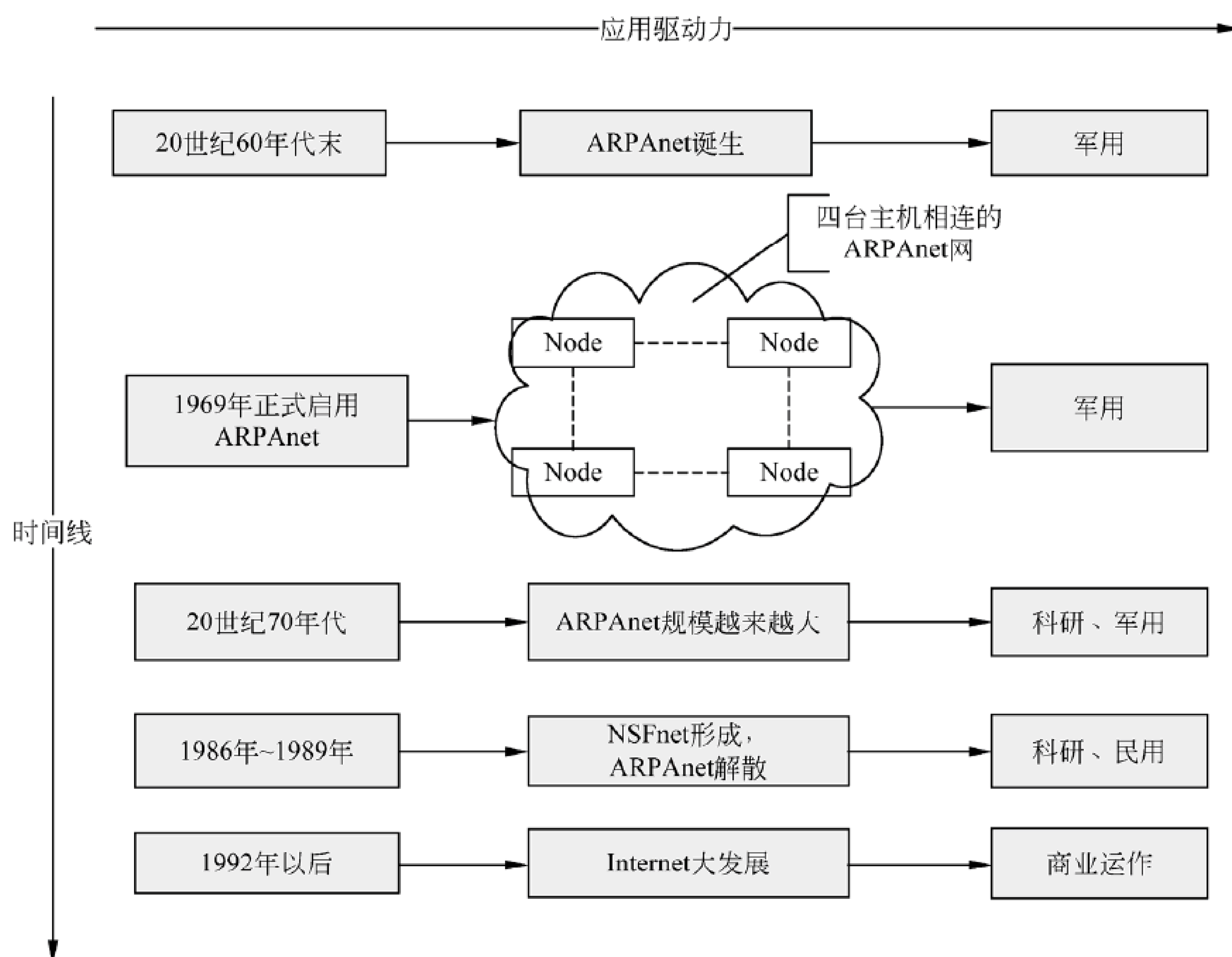


图 1.1　Internet 的起源与发展流程图

注意：ANS 的全称是 Advanced Network Service，高级网络服务的意思，在 Internet 建立之初，ANS 是整个 Internet 的重要组成部分。

今天的 Internet 已不再是计算机人员和军事部门进行科研的领域，而是变成了一个开发和使用信息资源的覆盖全球的信息海洋。基于 Internet 或嵌入式 Internet 的应用应有尽有，并且涉及各行各业，图 1.2 展示了与 Internet 有关的各种应用。

1.1.2　Web 的憧憬与发展瓶颈

Internet 的广泛普及和应用，标志着一个崭新的 Internet 时代的到来。将 Internet 带入新境界的就是被尊称为“万维网之父”的英国科学家蒂姆•伯纳斯。1991 年，在欧洲粒子物理研究所工作的伯纳斯为了高能物理研究的需要发明了万维网。万维网技术与 Internet 的完美结合，是 Internet 应用史上最辉煌的成就。

万维网（称作 Web、WWW、W3，英文全称为 World Wide Web），是一张附着在 Internet 上的覆盖全球的信息“蜘蛛网”，镶嵌着无数以超文本形式存在的资源，这些资源由一个全域的统一资源定位符（URL）进行标识，在使用的过程中，用户可以通过超文本传输协议（Hypertext Transfer Protocol）来访问这些资源，图 1.3 展示的是一个 Web 系统模型。

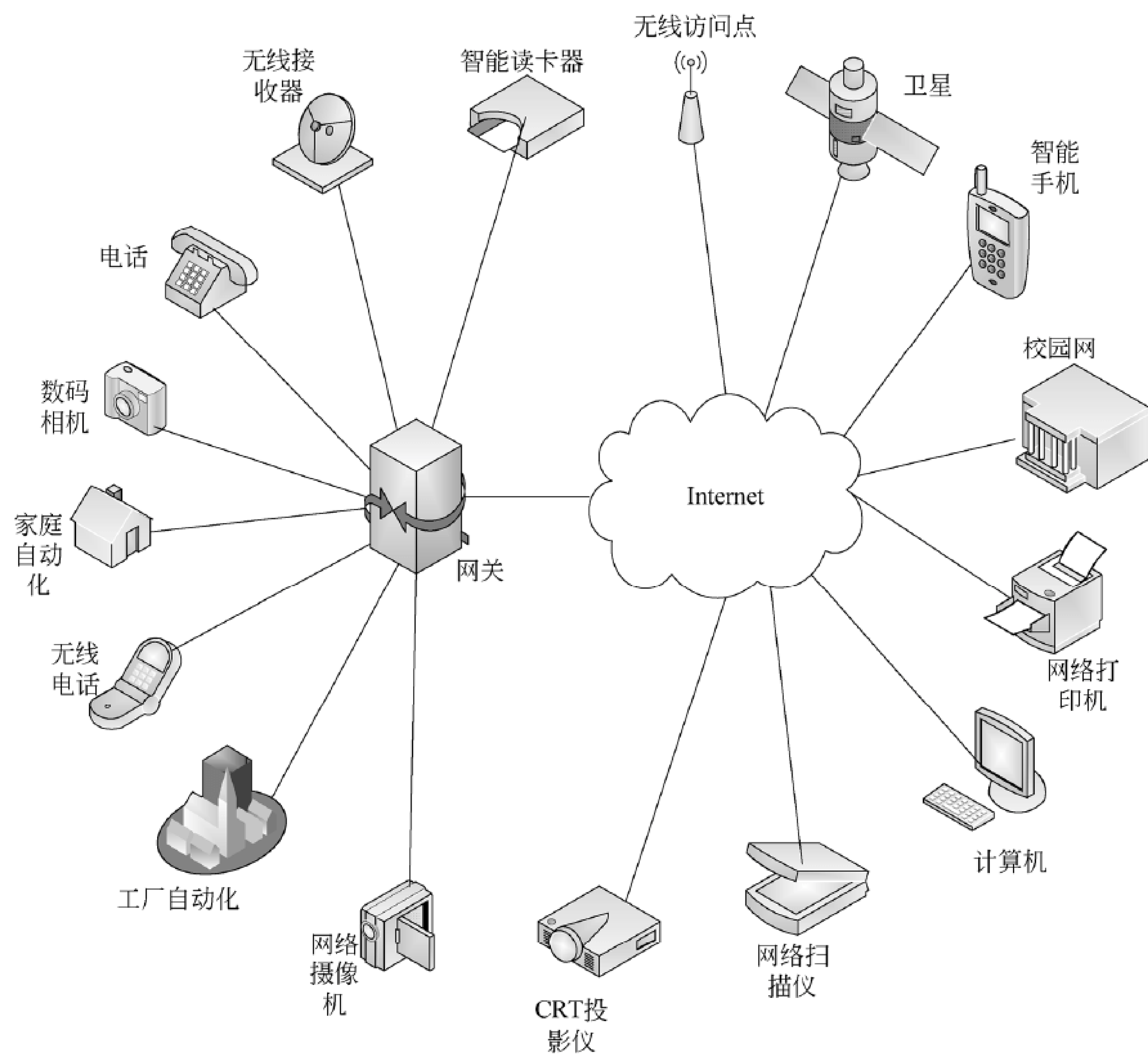


图 1.2　Internet 在各领域的应用

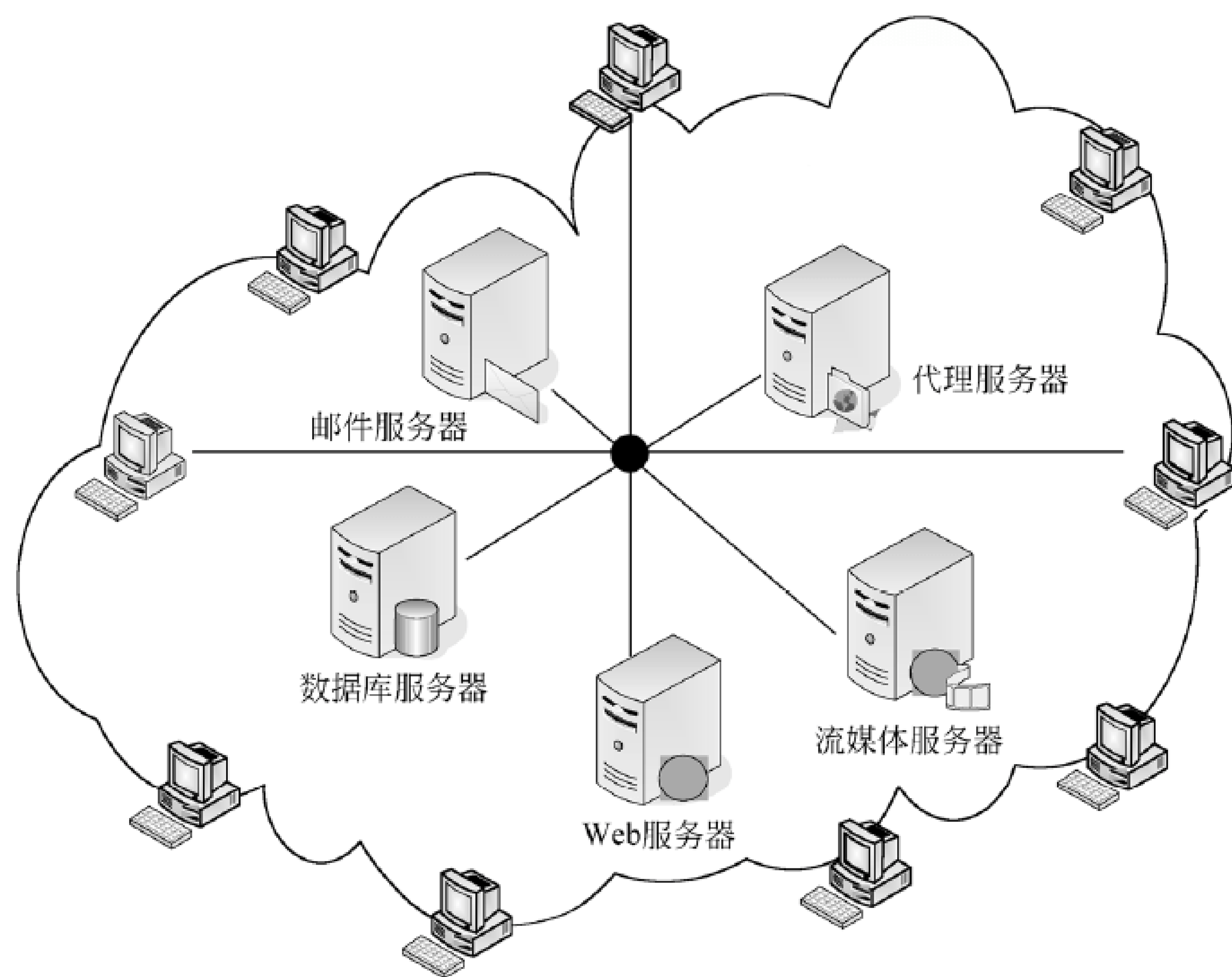


图 1.3　Web 系统模型

从图 1.3 中可以看出，整个 Web 系统的主干由各种不同的服务器组成，它们构成了整个 Web 网络服务的主体，用来向各类不同的主机提供不同的服务。各服务器的资源以 URL 进行标识和访问，资源以不同的协议进行传输，在用户一端，只需通过浏览器就可以轻松地访问到任何服务器的任何资源。

WWW 是 Internet 上最受欢迎、最为流行、最新的信息检索服务系统。其核心是 Web 浏览器和超文本标记语言（HTML，Hypertext Markup Language）。通过它，可以把 Internet 上现有资源统统连接起来，使用户能在 Internet 上已经建立了 WWW 服务器的所有站点提供超文本媒体资源文档。这无疑给全球信息的交流和传播带来了革命性的变化，一举打开了人们获取信息的方便之门。

注意：HTML 是用于描述网页文档的一种标记语言。在 WWW 上的一个超媒体文档称之为一个页面（page）。作为一个组织或个人在万维网上起始点的页面称为主页 Homepage，或首页，主页中通常包括有指向其他相关页面或其他结点的指针（超级链接）。在逻辑上将视为一个整体的一系列页面的有机集合称为网站（Website 或 Site）。网站可以用来发布网页，而网页的本质就是 HTML，通过结合使用其他的 Web 技术（如脚本语言、CGI、组件等），可以创造出功能强大的网页。因而，HTML 是 Web 编程的基础，也就是说万维网是建立在超文本基础之上的。

万维网的出现及其快速发展，使得全世界的人们以史无前例的巨大规模相互交流，各类 Web 站点、服务网站也以爆炸性速度增长，如图 1.4 是从 1991 年到 2007 年，全球范围内 Web 站点的增长情况，从图中可以很直观地看出万维网的发展速度。

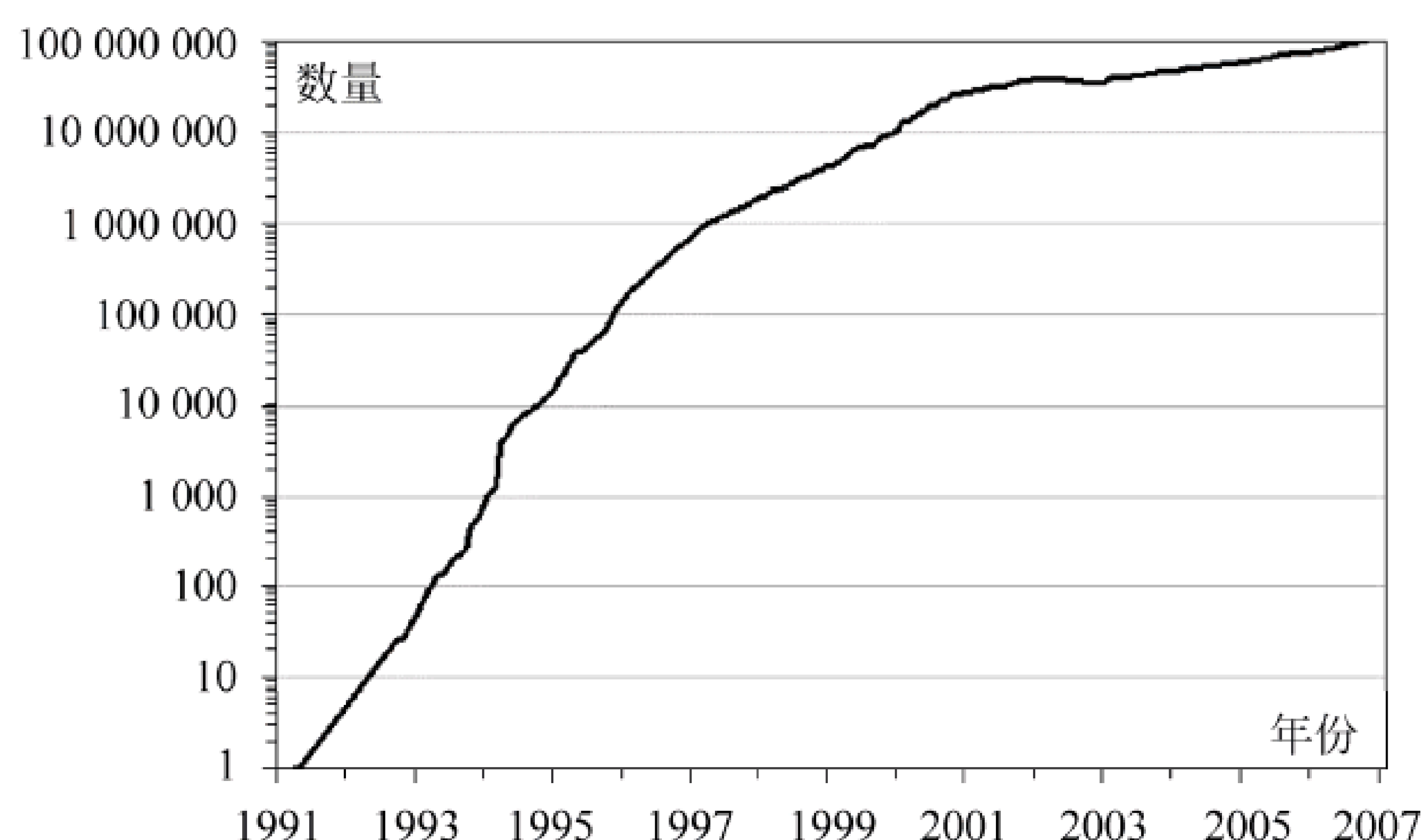


图 1.4　全球 Web 站点从 1991 年～2007 年数量增长趋势图

Web 的突出表现和它在全球范围内的高速扩张，使得其从诞生之日起便憧憬着创造一个共同的信息空间。在这个信息空间里，用户可以通过它实现平等的信息共享、信息交流、信息传输，可以用一个拓扑的结构的形式组织全球的资源以实现资源的充分利用和共享。当用户在网上进行互动活动时，电脑就会帮助用户分析这些活动，使用户明白自己在做什么，每个人在什么位置，以及如何更好地协同工作。

Web 的憧憬与梦想可以说是互联网的精神和核心价值，也是所有网络工作者孜孜以求的目标。但随着对 Web 的熟悉，人们开始发现，用户使用 Web 虽然可以容易地访问信息，但却无法很容易地发布信息，Web 的发展遇到了瓶颈。信息的接受者只要操纵一个鼠标，

便可以实现共用网的目的，而信息发布者却必须使用 HTML、DNS 注册、Web 服务器、公共网关接口（CGI，Common Gateway Interface）和一些冗长而乏味的缩写词和技术术语来实现信息或资源的发布。

如果信息共享和服务对等的互联网平衡被打破了，这种不平衡性实际上恰恰违背了互联网平等的概念，也正由于这种不平等的信息交流、网络中的对等共享、协同工作等其他功能的充分发挥受到一定的制约。

虽然人们依然喜欢浏览器，并经常惊叹于 HTML 页面的炫目图像，但普通网络用户使用最多的还是与他们生活关系最密切的电子邮件和在线聊天。而不可否认的是，电子邮件和在线聊天所使用的正是正宗的 P2P 应用程序，虽然这只是 P2P 功能的一部分，但足以让人们看到突破 Web 发展瓶颈的希望之光。

1.2　横空出世的 P2P

在 1.1 节中讲述了 Internet 的起源及 Web 思想的初衷，同时也说明了 Web 网络在实际应用中遇到的问题。那么，遭遇瓶颈 Web 网络要继续发展，要继续完成其平等、共享、协作的使命，就必须寻求新的突破口，而就在这个时候，P2P 技术横空出世了。本节将重点讲解 P2P 的概念及 P2P 思想的深层意思。

1.2.1　Peer-To-Peer 介绍

P2P，即 Peer-To-Peer 的缩写，Peer 一词在牛津英文词典上的解释如下。

Peer：a person of the same age，the same social position，or having the same abilities as other people in a group。

这段英文解释直译过来 Peer 的意思就是：同辈头衔、地位、能力或背景相同的人。两个 Peer 再加上一个连接词 To 就构成了 Peer-To-Peer，其意思就是伙伴对伙伴、对等、点到点的意思。将 Peer-To-Peer 的意思应用到互联网上就是对等互联的网络、点到点传输的网络，可以用图 1.5 来形象地表示 Peer-To-Peer 网络的结构。



图 1.5　Peer-To-Peer 网络形象结构图

在图 1.5 中，有很多手拉着手的小伙伴，他们组织在一起，形成了一个“伙伴式的网络”。在这个组织中，每个“伙伴”之间都是对等的，这是 Peer-To-Peer 的核心思想，也是对 Peer-To-Peer 网络最直观、最基本的解释。

目前，在学术界、工业界对于 P2P 都没有一个统一的完整定义，通常我们所说的 P2P 网络（以后文中所说的 P2P 都指的是 Peer-To-Peer）有下面两层意思。

（1）Peer-to-peer is a type of Internet network allowing a group of computer users with the same networking program to connect with each other for the purposes of directly accessing files from one another's hard drives。

这里的解释说的是：P2P 网络是一种用户之间通过某一相同的网络应用程序联系起来，彼此之间可以相互访问、共享计算机资源的网络。简单地说，P2P 就是一种网络概念。

（2）Peer-to-peer networking（P2P）is an application that runs on a personal computer and shares files with other users across the Internet. P2P networks work by connecting individual computers together to share files instead of having to go through a central server。

这种解释指的是：P2P 网络是一种不通过中央服务器而将一些独立的计算机资源组织起来，通过 Internet 运行于个人计算机上，以实现共享文件和资源的应用。这种解释将 P2P 定义为一种应用。

综合这两种解释就不难发现，其实 P2P 就是一种网络，一种架构在 Internet 上的网络技术。其核心思想是没有了中央服务器的概念，将 Internet 建立在对等互联的基础上以实现最大程度的资源共享。

1.2.2　P2P 的意义解析

互联网能够发展至今，根本原因在于其布建的任何一根血脉都是为人与人之间的交流而设置的。而现在能够引起互联网震动的，无非也只有交流方式的变革本身。如今，在基于网络的各种技术充斥于我们周围之时，恐怕只有很少人不知道 P2P 的概念了。上文已经对 P2P 的基本概念作了基本说明，也阐述了其直观的解释，关于 P2P 技术还有更深的思想内涵。

在具体讲解 P2P 的意义之前，先看一个小漫画，如图 1.6 所示。

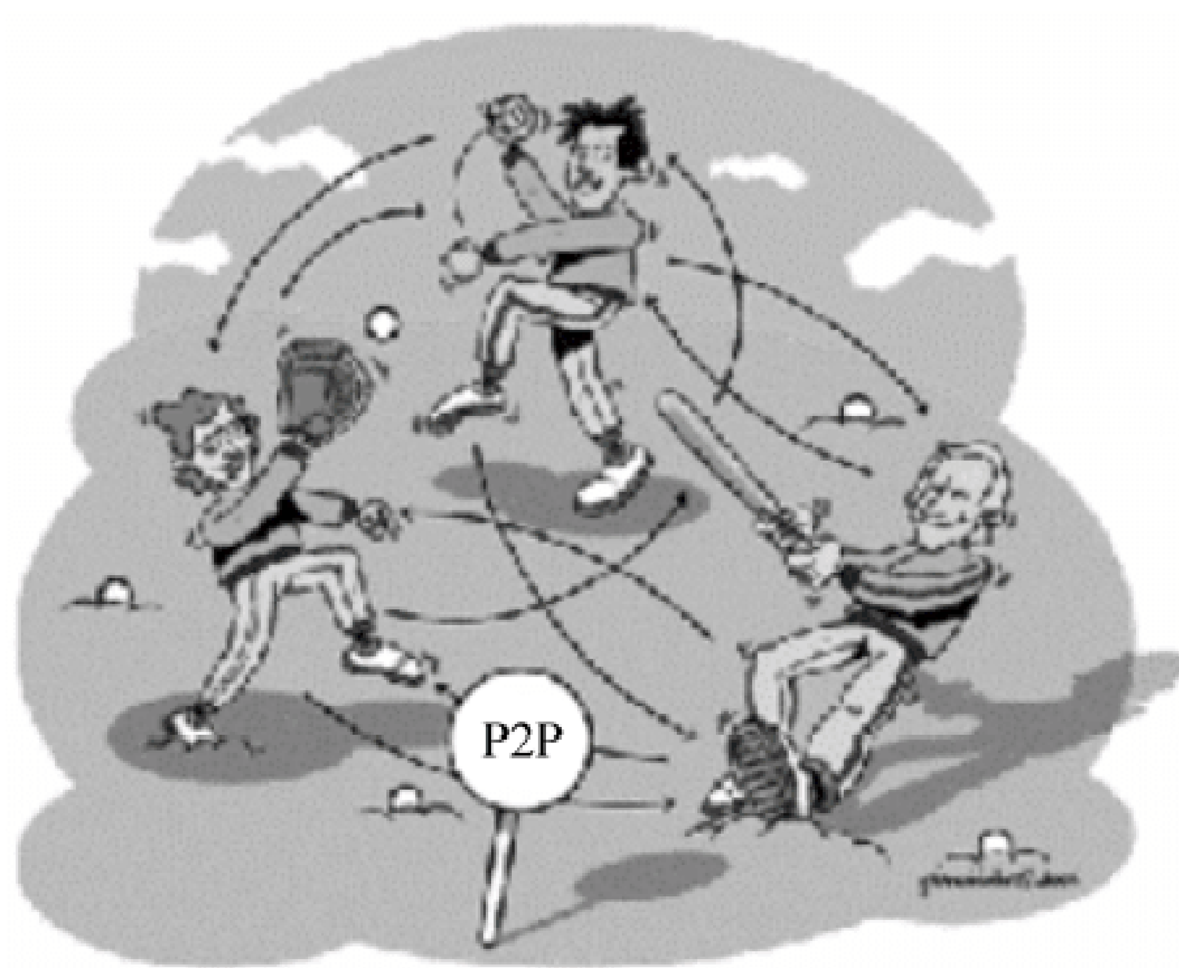


图 1.6　漫画 P2P

漫画 P2P 中，可以看到有 3 个人在玩一种名叫 P2P 的传球游戏，如果从漫画的角度来解析这个游戏的规则，那么这个 P2P 的游戏应该有这样几个特点：

- 组成这个游戏的成员，他们之间的关系是对等的，也就是 Peer-To-Peer 的意思，他们之间，不以任何人为中心，而是彼此独立的传球。
- 球是从一个人传到另一个人的，同一个人即担负着发球的责任，同时也有传球的义务。球的发出和接收，都由“人”来负责，参与游戏中的每一个人都有两种角色，发球者同时也是接球者。
- 球的传递是从一个点到另一个点的，如果脱离了这个点则球在本次的传递中就会丢失，在游戏的过程中，每一个点都可以添加一个新球进行传递。

以上 3 点是对游戏规则的描述，如果将其应用到 P2P 网络上，那么关于 P2P 的思想内涵也同样包含这 3 层意思。

- P2P 网络是对等互联的网络，组成网络的各结点之间是对等、伙伴式的关系。
- P2P 网络结点之间资源的传递是从一点到另一点的，每个结点既是资源的提供者同时也是资源的发布者。
- P2P 网络中的资源是依赖于结点而存在的，且资源可以动态地加入或退出。

以上是对 P2P 意思的进一步分析，认识 P2P 只理解其字面意思是不够的，更应该明白一点的是，P2P 是一种思想，一种有着改变整个互联网基础潜能的思想。客观地讲，单从技术角度而言，P2P 并未激发出任何重大的创新，而更多的是改变了人们对因特网的理解与认识。正是由于这个原因，IBM 早就宣称 P2P 不是一个技术概念，而是一个社会和经济现象。

注意：拿破仑有句名言“世界上只有两种强大的力量，即刀枪和思想，而从长远来看，刀枪总是被思想战胜的。”这足以证明思想的力量之大。理解 P2P，不仅要理解它的概念、特点和意义，更要理解它所蕴含的思想。

不管是技术还是思想，P2P 是直接将人们联系了起来，让人们通过互联网直接交流。它使得网络上的沟通变得更容易、更直接，真正地消除了中间环节。

从 P2P 所体现的思想来看，它是一个全新的概念，因为它可以改变现在的 Internet 中以服务器、集群、大网站为中心的状态，重返非中心化、分布式的结构，并把权力交还给用户，让我们的语言、影像、资源、信息等以最直接、最快捷的方式传递到对方身边。它最符合互联网络设计者的初衷，给了人们一个完全自主的超级网络资源库，真正实现了人人为我，我为人人的网络里的“大同世界”。

1.2.3 P2P 的定义和特点

上文中对 P2P 所蕴涵的几层意思进行了解析，下面给出 P2P 数学意义上较严格的定义。

- 有某网络 N，是架构在 Internet 之上的网络，满足 Internet 网络的所有基本特征。
- 在 N 网络中，存在两种基本的行为模式，一种行为定义为 P，是生产资源（提供资源）的行为，另一种行为定义 C，是消费资源（接受资源）的行为。
- 组成 N 网络的所有网络结点（Peer）之间是对等的关系，且同时具备行为 P 和行为 C。

- 在 N 网络中，各 Peer 之间以无中介的、对等的方式进行双向交换，以执行 P 和 C 的功能。
- N 网络依赖于 Peer 的存在而存在，且 Peer 可以自由地加入或退出。
- 当有 Peer 加入或退出时，N 仍保持组织、结构等特性不发生改变。

满足以上 6 个条件的网络 N，就可以说是 P2P 网络。在这个网络中，实现了资源的生产与消费的对等平衡，实现了信息和服务在一个个人或对等设备与另一个个人与对等设备间的双向流动。一个完整的 P2P 网络模型如图 1.7 所示。

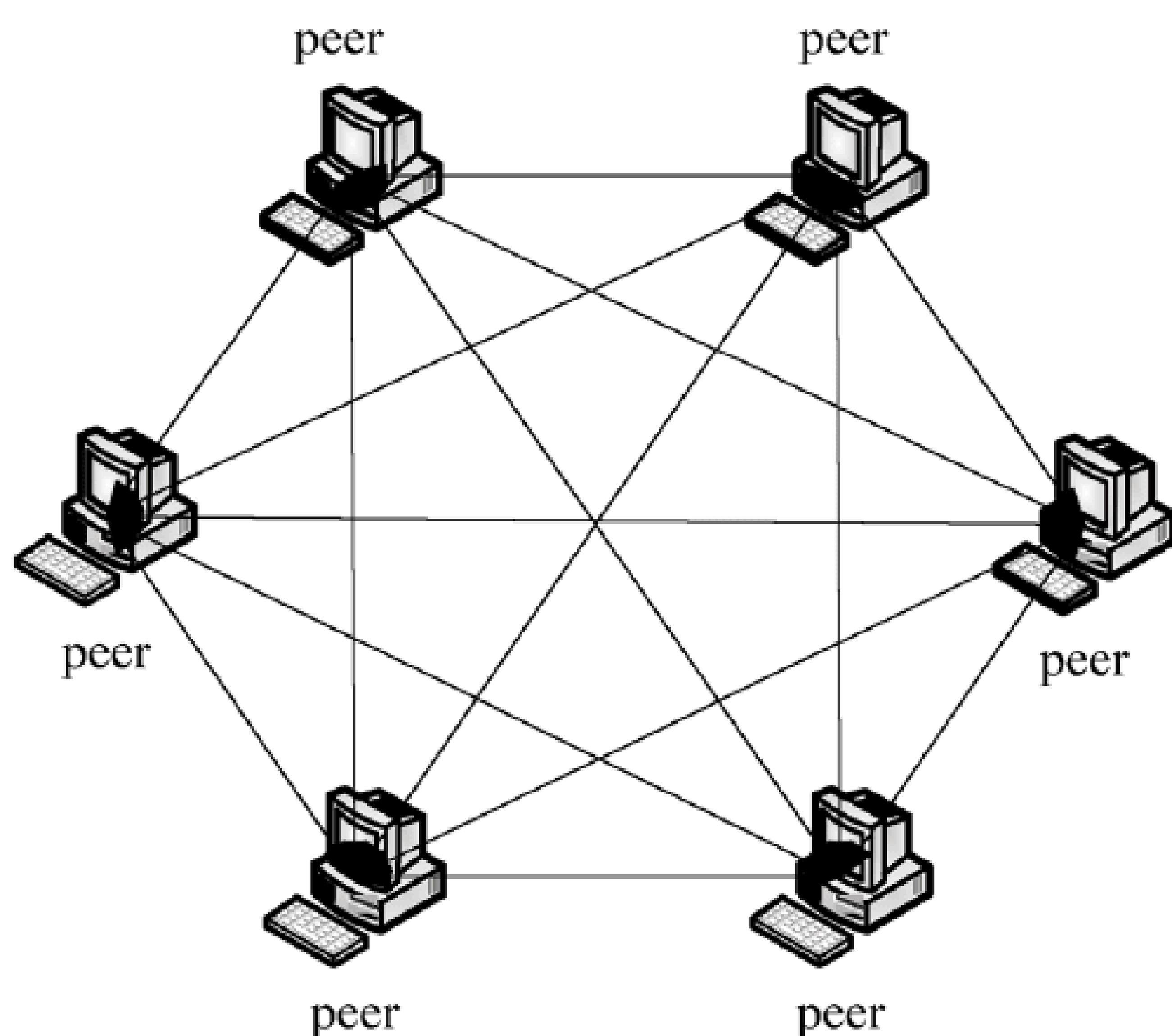


图 1.7　P2P 网络结构模型图

根据 P2P 的定义可知，一个 P2P 网络应该具备以下特点。

- P2P 是对等的：资源的发布与接受两个角色合二为一，在生产和消费资源的角色上是对等的。
- P2P 是直接的：在 P2P 网络中无中介、等级、格式、区域和平台的限制，相互之间直接交换信息和服务。
- P2P 是动态的：组成 P2P 网络的结点可以动态地加入或退出，在运行过程中也是动态的提供资源和服务。
- P2P 是双向的：P2P 网络中，结点之间是最直接、最纯粹的双向关系，切切实实地实现了资源和服务的交换与共享。
- P2P 是及时的：无服务器参与空间分配，可提供实时的、可升级的信息。
- P2P 是有效的：可充分利用个人计算机的软硬件设备，信息和服务在结点间传输时交互的对象及接收的目标是确定的、有效的。

这就是 P2P。它以用户为中心，所有的用户都是平等的伙伴。相隔万里的用户可以通过 P2P 共享 PC 机上的文件、目录乃至整个硬盘。所有人都自由地分享他们认为最有价值的东西，P2P 使互联网上信息的容量、范围、价值得到极大的提升。P2P 技术带来的这种用户间直接交流的方式，真正实现了互联网共享和自由的梦想，它改变了互联网现有的游戏规则，也改变了我们的生活。

1.3　P2P 与 Web 的对比与较量

上文对 P2P 的概念及深层意义都进行了相应的讲解，如果独立地去解释那些纯粹的概念，并不能说明 P2P 就是互联网上一场新的变革。从 ARPAnet 到 NSFnet，从 NSFnet 到 Internet，以 Web 技术引领的互联网络改变了世界，传统意义上的 Web 网络模型已深入到了现实社会并深刻影响和改变着普通人类的生活。在这种背景下，P2P 成长起来而且呈现出巨大的活力和潜力。要进一步了解 P2P，就要知道 Web 与 P2P 的异同，本节的重点，就是对 P2P 和大家熟知的 Web 做一个比较说明。

1.3.1　基于 Web 的 S/C 与 P2P 在结构上的比较

目前，Web 网络的主要技术模式是 S/C 方式的，关于 S/C 模式的网络结构模型，如图 1.8 所示。

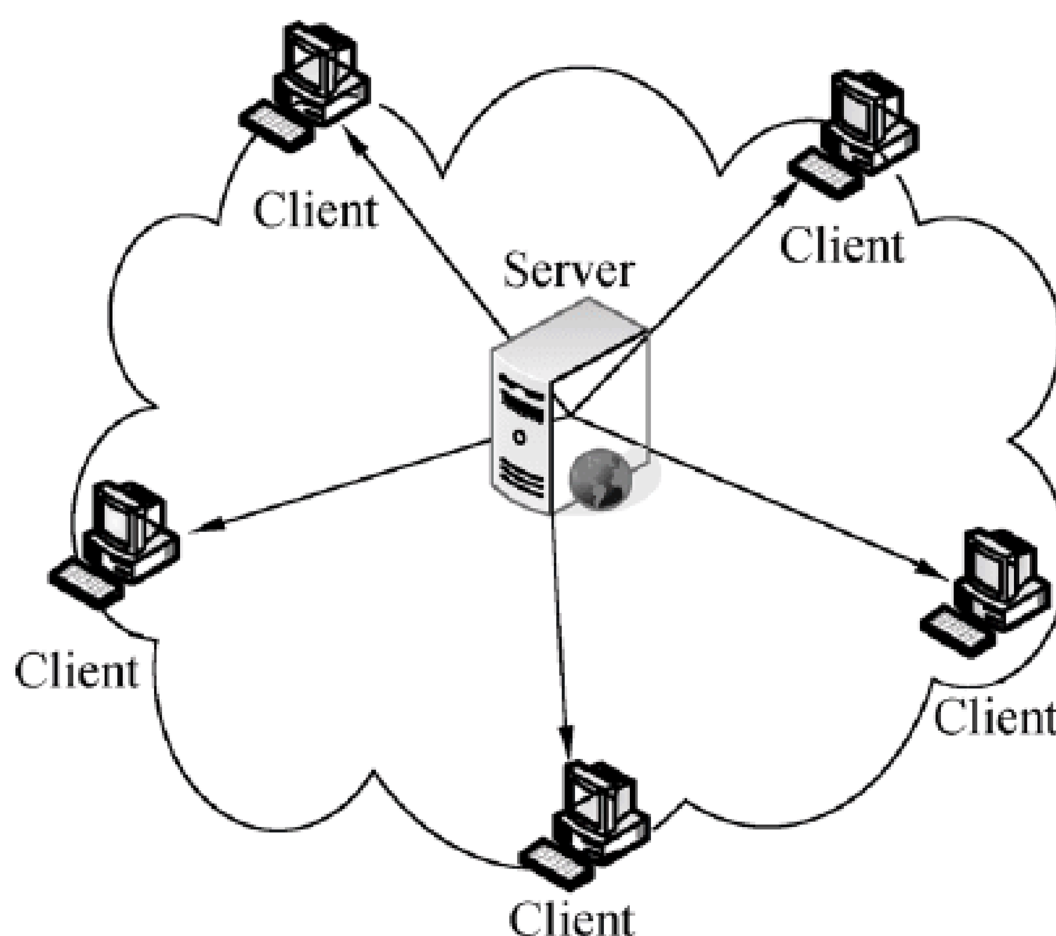


图 1.8　基于 Web 的 S/C 网络结构模型图

图 1.8 中，位于中心位置的就是作为 Server 的服务器端。在服务端，作为中央服务器的主机一般都是有强大处理能力和大带宽的高性能计算机。中心服务器在集中处理数据的同时还要有对互联网上其他 PC 提供服务。对于一台与服务器联机并接受服务的 PC 来说，这台 PC 就是客户机。客户机有很多，可以同时与服务器端相连，在性能要求上可以相对弱小，这种服务器（Server）与客户端（Client）组织成的网络模式就是 S/C 模式。

注意： 实际应用中，在基于 S/C 模型的 Web 网络中要使中央服务运行起来，还要配合高档的服务器软件，如 Web 服务、数据库服务、文件服务等，再将大量的数据集中存放在这个服务器上，这样位于客户端的机器才可以访问。

与 Web 技术不同的是，P2P 技术打破了传统的 Client/Server（C/S）模式，其技术的特征之一就是弱化了服务器的作用，甚至取消服务器，在网络中的每个结点的地位都是对等的。每个结点既充当服务器，为其他结点提供服务，同时也享用其他结点提供的服务。P2P 与 C/S 在不同方面的特性可以通过表 1.1 来说明。

表 1.1　P2P模式与C/S模式的网络特性比较

网络特点	P2P 模式	C/S 模式	网络特点	P2P 模式	C/S 模式
数据分发	好	差	动态性	好	差
数据传输	中	好	自组织性	好	差
数据交互	好	差	可扩展性	好	差
数据容量	好	差	抗干扰性	好	差
数据即时性	好	差	安全性	差	好
数据质量	中	好	易管理性	差	好
数据覆盖率	差	好	成本控制	好	差
容错性	好	差			

通过表 1.1 可以清楚地看到，P2P 模式和 C/S 模式是各有优劣的，不同的网络特性对应着不同的应用，在选择网络模式时可根据应用的侧重点不同选择不同的模式。

1.3.2　Web 与 P2P 在资源传输上的比较

在 Web 网络中，当用户间要进行信息资源的传输活动时，首先要构建一个有一定资源的站点，然后在其他 PC 上创建信息并“发布”到站点上。这些信息在站点上等待请求。接收到请求之后，站点将信息传递给请求者。整个过程需要 3 步，即创建资源——发布资源——接收资源。

同样是进行信息资源的传输，在 P2P 中则不同。处于对等地位的设备的各个 PC，可以直接向存储着源文件的另一对等设备（PC）发送请求。而接收到请求的 PC，无须经过第三方中介，可以直接将需要的信息发送给对方。整个过程只需两步，即创建资源——接收资源。

如图 1.9 所示，若客户机 Client B，要请求另一对等客户机 Client A 的资源，以 Web 的方式和以 P2P 的方式有根本的不同，如下所示。

（1）Client B 以 Web 的方式查看 Client A 上的资源时：

- 客户机 Client A 与 Web Server 建立连接，将创建的资源发布到 Web Server 上。
- 客户机 Client B 与 Web Server 建立连接，并向 Web Server 发送查看资源的请求。
- Web Server 将 Client A 发布的资源以 Web 服务的形式提供给 Client B。

（2）Client B 以 P2P 的方式查看 Client A 上的资源时：

- Client B 直接与 Client A 建立连接，Client A 根据 Client B 的请求，无须经过第三方，可直接以 P2P 的方式将资源提供给 ClientB。

通过上面的比较可以得出这样的结论：在 P2P 中，信息的交换不需要通过第三方的中心站点，而直接在对等设备之间进行，（这种对等设备，在大多数情况下，就是在 Web 中位于 Internet 边缘的个人电脑），是一种高效直接的传输方式。这个结论还可以通过 P2P 和 Web 站点之间要素的比较得到进一步的说明，如表 1.2 所示。

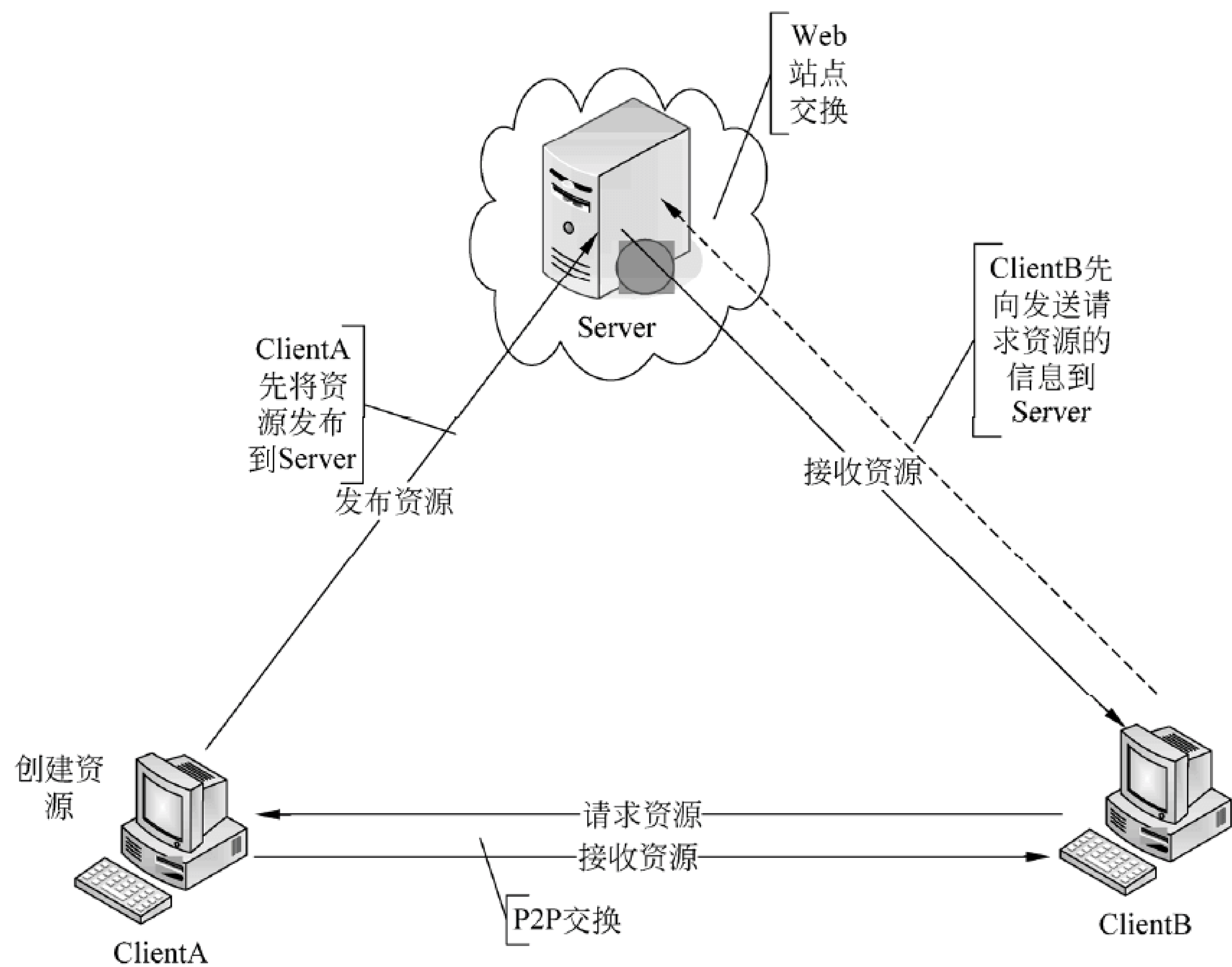


图 1.9　Web 站点交换与 P2P 传输的比较

表 1.2　P2P和Web站点之间的要素比较

要　　素	P2P	Web
资源交互对象之间的关系	对称	不对称
用户数量	越多越好	有上限值
内容和服务的提供	简便	复杂
网络可扩展能力	无限	有限
隐私的保护能力	较弱	较强
资源的数量	无上限	有上限值
交互的系统	有特定的客户端系统	通用的浏览器
请求的类别	大量不同动态的请求	大量相似的请求
运行结构	完全分布式	客户端/服务器
资源的分配	分布式	集中
文件格式	任意	基于 Web 传输协议的文件类型
机器类型	任意	大型服务器

从表 1.2 中可以看出，P2P 改变了 Web 下存在的信息消费者和生产者之间的不平衡，它允许更多的用户参与交换，而交换内容的过程和服务提供的方式都更加简便、直接；大量的信息不再集中于某个站点，而是被分布在多个对等设备中，这样，即使其中的一部分设备停止工作，其他参与其中的设备仍然可以完成传输任务。这种思想正是 Internet 创建的初衷所追求的。

1.3.3　Web 与 P2P 在请求方式上的比较

在基于 Web 的 S/C 结构中，对服务、资源的请求方式基本上是一站式的。客户端向服务器请求一个文件时，直接通过统一资源定位符（URL）就可以将请求信息发送给服务器，服务器在接到请求后进行相应处理，并将处理结果直接返回给用户，这种请求方式是简单的一问一答，中间过程不发生请求的转发或是目标的跳转。如图 1.10 所示为在 S/C 结构中，Client A 向远程服务器 Web Server 请求一个 Picture 文件的过程。

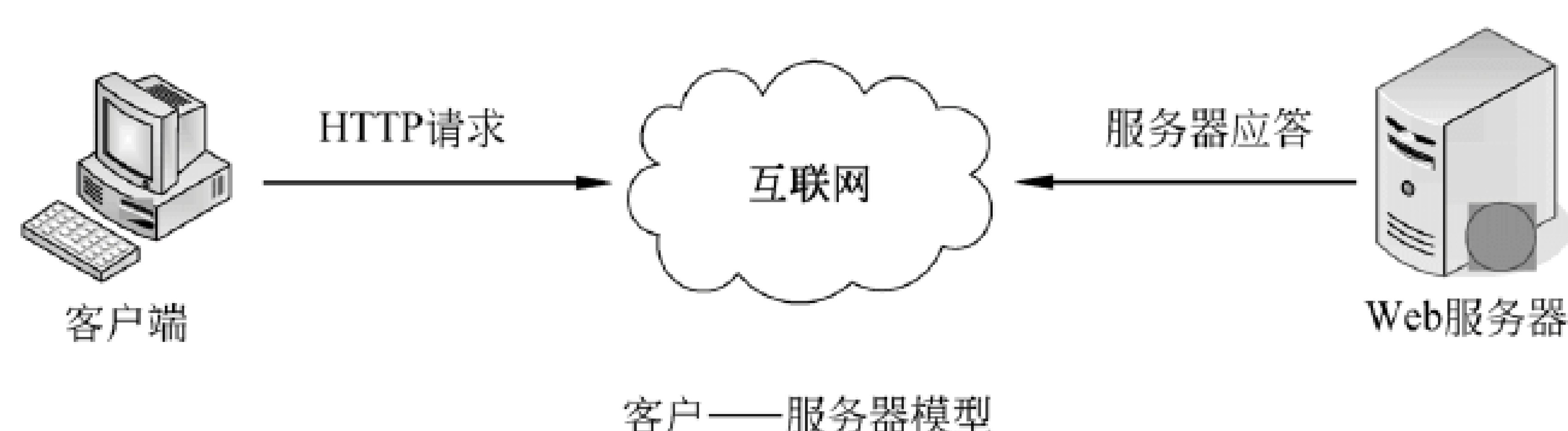


图 1.10　Web 中的 S/C 模式对资源的请求过程

而在 P2P 网络中，对资源或是服务的请求则是多站式的。当需要发送请求时，Peer A 会向所有与它相连的其他 Peer，如 PeerA1，PeerA2…Peer Ax…等，发送同样的请求。如果被请求的 Peer 中的一个 Peer Ax 中拥有被请求的资源，那么 Peer A 就与 Peer Ax 直接建立连接进行资源传输；如果没有被请求的资源，则 Peer Ax 会将此请求转发给与自己相连的其他 Peer，依次传递。请求信息会根据不同的搜索策略和算法规则在 P2P 网络中持续转发下去。如图 1.11 所示为 P2P 网络中结点对资源的请求方式。

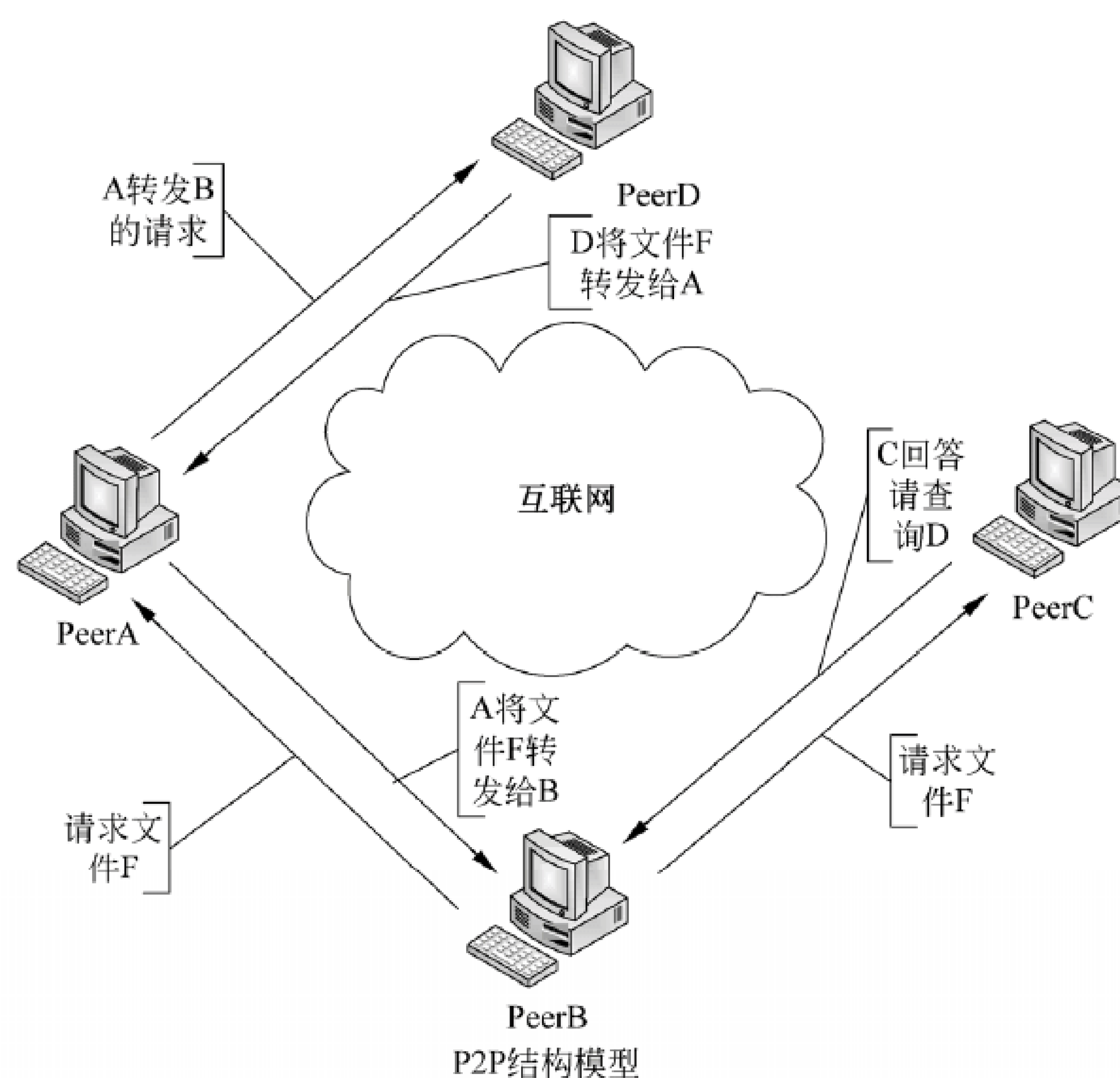


图 1.11　P2P 网络中结点对资源的请求方式

注意：不同的 P2P 网络结构对资源的请求有不同的方式，这也是在后文要讲到的 P2P 网络搜索技术。上面提到的此类请求方式，是针对通用意义上的 P2P 网络的。

1.3.4　竞争还是互补

基于 Web 的 S/C 模式造就了互联网的辉煌时代，然而，S/C 方式的结构特点和其内在的本质特性造成了互联网络上资源和服务的有向集中，无论信息资源还是成本资源均向同一方向集中。这种对资源的“中央集权”式的管理虽然与互联网的思想有些出入，但这样的模式符合一对多、强对弱的社会关系形式，如政府对个人、对企业，大企业对小企业，学校对学生，企业对职工等关系。所以 S/C 方式是符合市场需求的，在可预见的将来必将还有更广、更深的发展。

作为近几年成长起来的 P2P 技术，不论是 P2P 的结构方式还是其表达的互联网思想，都是最大限度的将信息数量、成本资源向互联网各点均匀分布，也就是所谓“边缘化”的趋势。此模式符合“一对一”的特点，也符合彼此相当的社会关系形式，如个人对个人，集团对集团，规模相当的企业之间等，这也是符合市场需求的。所以 P2P 与 Web 这两种方式不是你生我死的竞争关系，而是相互依赖、相互补充的共存关系，有关 P2P 即将替代 Web 网络上 S/C 模式的说法是没有事实根据的。P2P 有其独特的市场空间，是现有互联网应用的补充，这一点应该是毫无疑问的。

1.4　P2P 的起源与初步发展

P2P 从诞生到发展为今天的蓬勃兴起，经历了一个既快速又曲折的发展过程。了解 P2P 的发展与演进也就了解了 P2P 技术从孕育到萌芽到发展的步骤，对 P2P 技术的原理、特性及其面临的问题也都会有进一步的理解。本节将带领读者回顾 P2P 的历史，同时讲述 P2P 在发展过程中的标志性事件、P2P 系统的典型代表，以及 P2P 技术在发展过程中所遭遇的喜怒哀乐，进一步理解 P2P 技术的发展之路。

1.4.1　从 ARPANET 追溯 P2P 的起源

P2P 的思想及其深层意义在前面已经有过讲述，然而这一思想并不是一项新的技术，追溯 P2P 的起源可以从 20 世纪 60 年代后期 ARPANET 的建立开始。ARPANET 建立的最初目标是共享在不同美国研究机构间的计算资源和文档，在这个原始系统中，并没有现在网络中的客户机/服务器这种模式，每台主机互联，相互之间都被同等地对待，这是 P2P 思想的核心，虽然这个网络并不是自组织的，也没有完整的体系可言，但是，可以说从 ARPANET 诞生那一刻开始，最早的 P2P 思想雏形也随之出现。

1990 年前后，是 Internet 的大发展时期，在互联网世界出现了许多应用，如 WWW，电子邮件和流化服务等，这些应用都是建立在客户/服务器通信模型的基础上。在这种模型中，有若干中央服务器处于网络的中心地位，而与之连接的客户机则处于网络的边缘地位，

在客户端一侧，用户通过发送特定的请求来访问服务器一侧的各种资源，网络行为都是建立在客户与服务器之间的关系上，客户与客户之间则基本是“绝缘”的。直到 Napster 的出现，一种新的基于 P2P 的关系也正式进入互联网了。

1.4.2　P2P 的萌芽

USENET 的出现可以说是 P2P 的萌芽时代，它产生于 1979 年，是一个巨大无比的网上讨论组，一般也称为“新闻组”（newsgroups）。你可以将它想象成一个包罗万象、无所不有的网上论坛，但是它又不同于普通的论坛。

20 世纪 70 年代末，当时还没有互联网和浏览器，它们都要在十多年后才会出现。那时所谓“上网”，就是用 modem 拨一个电话号码，连到另一台机器，收收邮件，看看主机上面系统管理员发的通告。如果想换一台主机看看，那就必须先挂断，然后再拨另外一个电话号码。这样的上网方式很不利于开展多人的讨论，因此基于当时的需求，迫切需要一种大规模的、分布式的、多中心的远程信息交换手段。

1979 年，Duke 大学的两个研究生 Tom Truscott 和 Jim Ellis，提出一种分布式网上讨论组的构想，根据这种设想，他们创建了一种分布式远程信息交换和讨论的系统，这种系统也叫讨论组。在讨论组创建之初，主要是供 UNIX 爱好者协会（USENIX）的成员使用，因此就被定名为 USENET。如图 1.12 所示为 USENET 的系统结构。

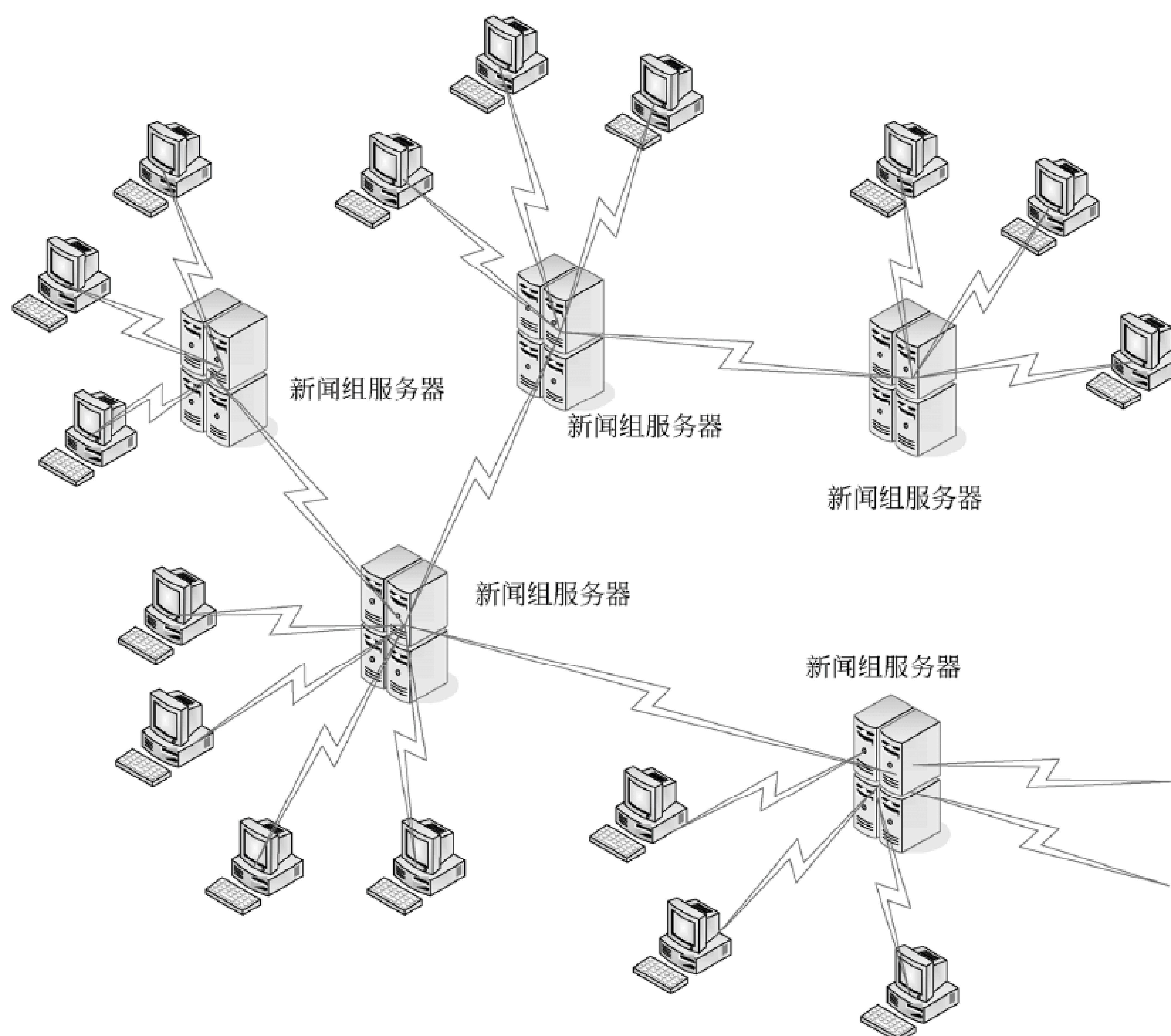


图 1.12　USENET 系统结构模型图

USENET 的运行机制其实非常简单。对于用户来说，只有 3 步。

（1）网络服务提供商在一个网络中，设定一台服务器作为 USENET 专用服务器，再将它的网址告诉用户。

（2）用户想要发言的时候，就向这个网址发送帖子（post），这与发送 E-mail 很相似，但是两者格式不一样，在 USENET 上发言必须使用专用的客户端。

（3）查看其他人的发言时，就必须从服务器上下载其他人的帖子。下载完成后，如果想回复某人的帖子，就再重复第 2 步。

USENET 服务器每天会同其他 USENET 服务器交换帖子。这就是说，全世界所有的 USENET 服务器最终都可以互相交换帖子，保持内容的同步。所以理论上，不管你的帖子是发到哪一台服务器上，最终全世界的人们都会看到，并且会从世界各地给你回复。可以看到，在这种应用中，系统可帮助用户组织内容并提供一种自组织方法，由参与用户通过一个严格的执行过程来添加或去除新闻组服务器，这一过程初步体现了 P2P 思想的端倪。

FidoNet 是继 USENET 之后基于 P2P 思想的又一典型应用，它是由 Tom Jennings 于 1984 年创建的，是早期 P2P 应用的杰出代表。它和 USENET 类似，也是一个分散、分布的信息交换系统，通过 FidoNet 系统，可以让不同 BBS 系统中的用户们互相交换信息，不管是它的组织方式还是信息交换方式，都是 P2P 思想的体现。

USENET、FidoNet 都是一个分散、分布的信息交换系统，也都是值得探究的系统，因为它们在多年前就遇到并解决了许多当今 P2P 技术所面临的同样问题。这两种非常成功的分布式对等网络技术可以说是 P2P 技术的萌芽，在最初的 P2P 应用出现时，许多使用该技术的人们甚至不会使用计算机。然而正是这种孕育着思想的网络技术为 P2P 的出现搭建了摇篮。

1.4.3　P2P 的起步

P2P 正式步入发展的历史可以追溯到 1998 年，当时，美国波士顿大学的一年级新生，18 岁的 Shawn Fanning 编写了一个在网上搜索并下载音乐的简单程序，这个程序就是后来风行全球的 Napster。

Napster 是一款可以在网络中下载自己想要的 MP3 文件的软件，它同时能够让自己的机器也成为一台服务器，为其他用户提供下载。在这个网络中，Napster 本身并不提供 MP3 文件的下载，它实际上提供的是整个 Napster 网络的 MP3 文件“目录”，而 MP3 文件分布在网络中的每一台机器中，随时供你选择取用。在下载的时候并没有中央资源服务器的概念，而是直接连到另外一台对等的 PC 上，相互之间进行资源的交互和传输。在 Napster 中搜索下载音乐的过程可用图 1.13 来表示。

在图 1.13 中，当要找一个名为 debaser.mp3 的音乐文件时，需要下载此 MP3 文件的主机，先向中心服务器发出下载此文件的请求，在索引服务器上搜索此文件，然后服务器会根据其他从 Napster 客户端的注册信息，返回给需要下载的主机存储着此 MP3 文件的另一个 Napster Client 的信息，根据这些信息就可以直接与此 Client 进行通信，这样就能完成文件的下载。

通过这一系统，可以自由搜索想下载的 MP3 音乐文件，也可以将自己的资源共享出去，供其他需要的人下载，同时还可以进行多人聊天、在线论坛中进行讨论，互相交流等。Napster 令无数散布在互联网上的音乐爱好者美梦成真，可以不受版权的限制自由下载音

乐。无数人在一夜之内开始使用 Napster。有统计数据显示，在最高峰时 Napster 网络有 8000 万的注册用户，这是一个让其他所有网络应用系统望尘莫及的数字。这大概可以作为 P2P 软件成功进入人们生活的一个标志。

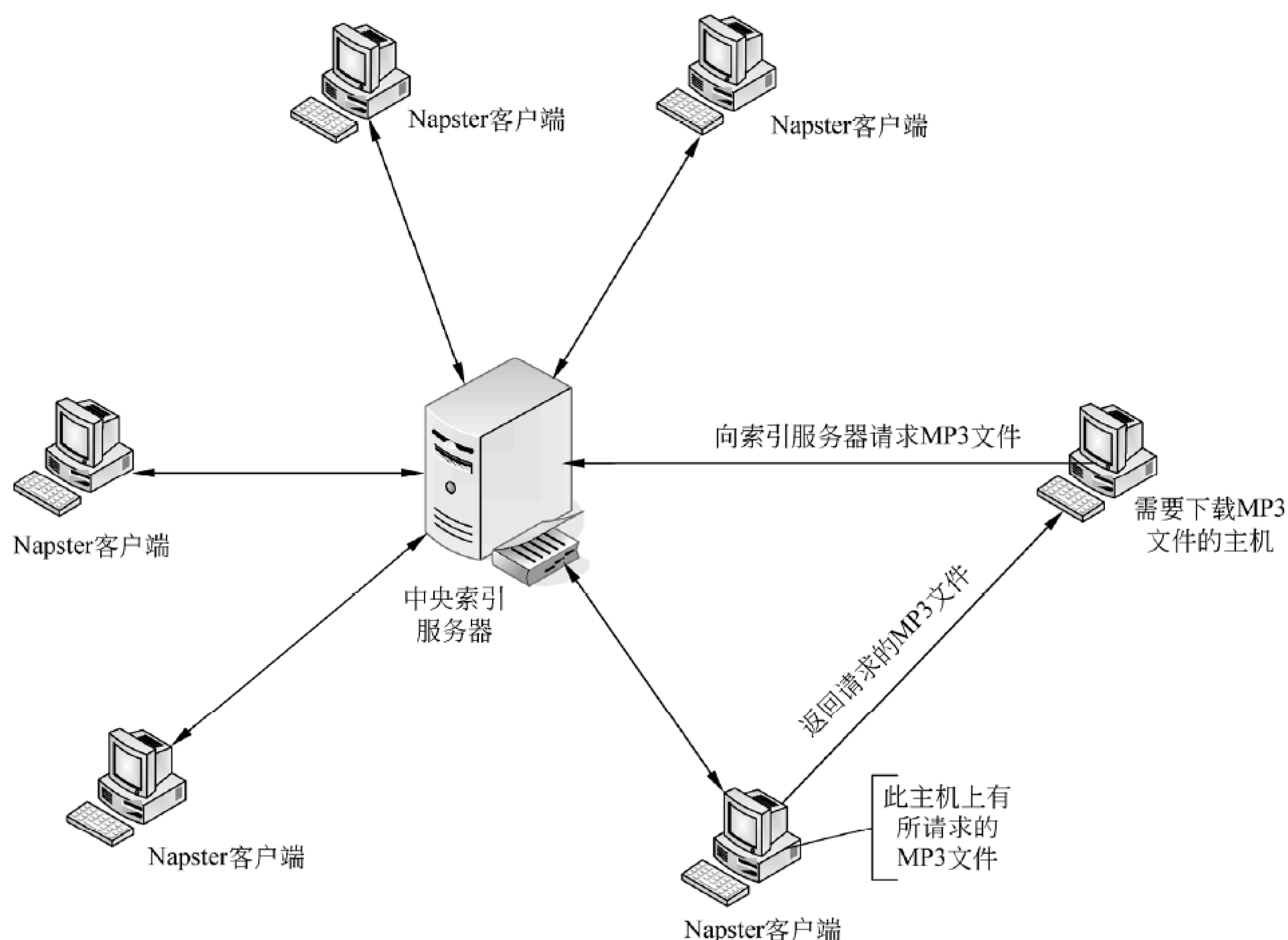


图 1.13　Napster 系统中的文件搜索模型图

Napster 系统有一个中心服务器，这一服务器并不是传统的 Web 服务器，它不对外提供应用服务，也不存放任何实体资源，只是作为实体资源的目录或索引而存在，因而也称为目录服务器。它存放所有文件的元数据信息，如文件的标题和一些简单的描述信息，以及拥有此文件的结点机器的 IP 地址。Napster 的系统结构如图 1.14 所示。

在这个系统中，中央服务器的作用只用来存储索引信息，当有新的结点加入系统时首先要连接目录服务器并报告自身地址及共享的文件列表。用户需要某个文件时向目录服务器提交搜索请求，目录服务器返回符合搜索要求的所有文件的存储地址，之后用户根据对应地址直接从共享此文件的结点处进行文件下载。由于目录服务器只提供索引服务，而不承担文件存储和下载服务，因此它支持上万结点同时在线。Napster 在发布后迅速流行起来，很快成为增长最快的网络应用系统。如图 1.15 是 Napster 的系统界面。

Napster 作为一个文件共享系统，解决了两个核心的问题：一是用户必须知道哪些机器上有哪些文件，这样当用户提出文件搜索请求时才可以得到正确的匹配结果。二是，当知道这个文件以后，如何与提供此文件的结点机器建立互联以实现下载。

Napster 作为一个纯粹的基于 P2P 技术的文件共享系统在初期取得了巨大成功，第一次验证了 P2P 思想在广域网范围内的可行性，也证明了 P2P 技术的优越性和巨大应用前景。Napster 的巨大成功，直接促成了 1999 年 5 月 Napster 公司的成立，这是 P2P 历史上重要

的开端，之所以说它是开端，是因为这是一个非同意义的起始，也正是从这天起，P2P 开始了它曲折但极富生命力的发展。

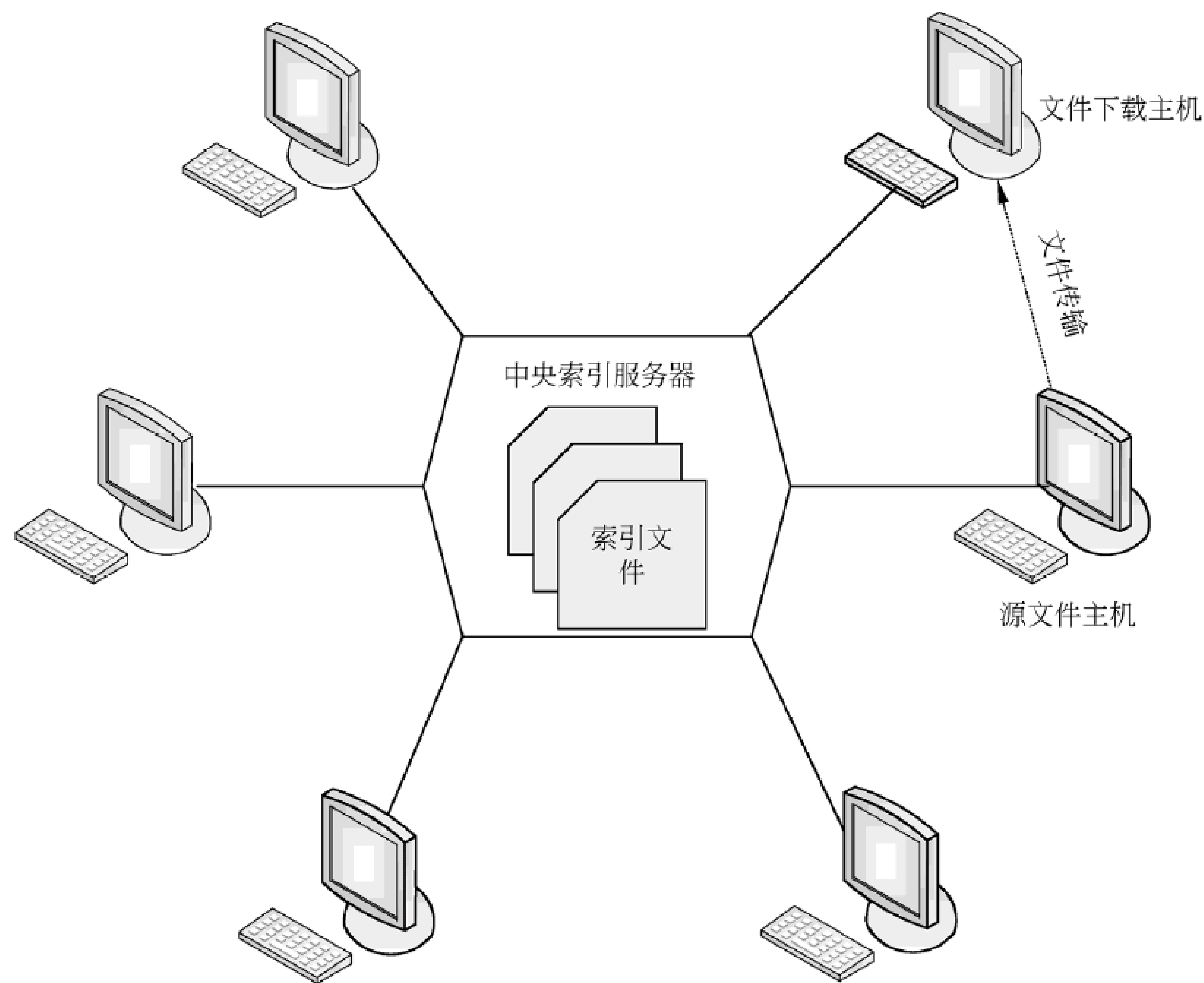


图 1.14　Napster 的系统结构图

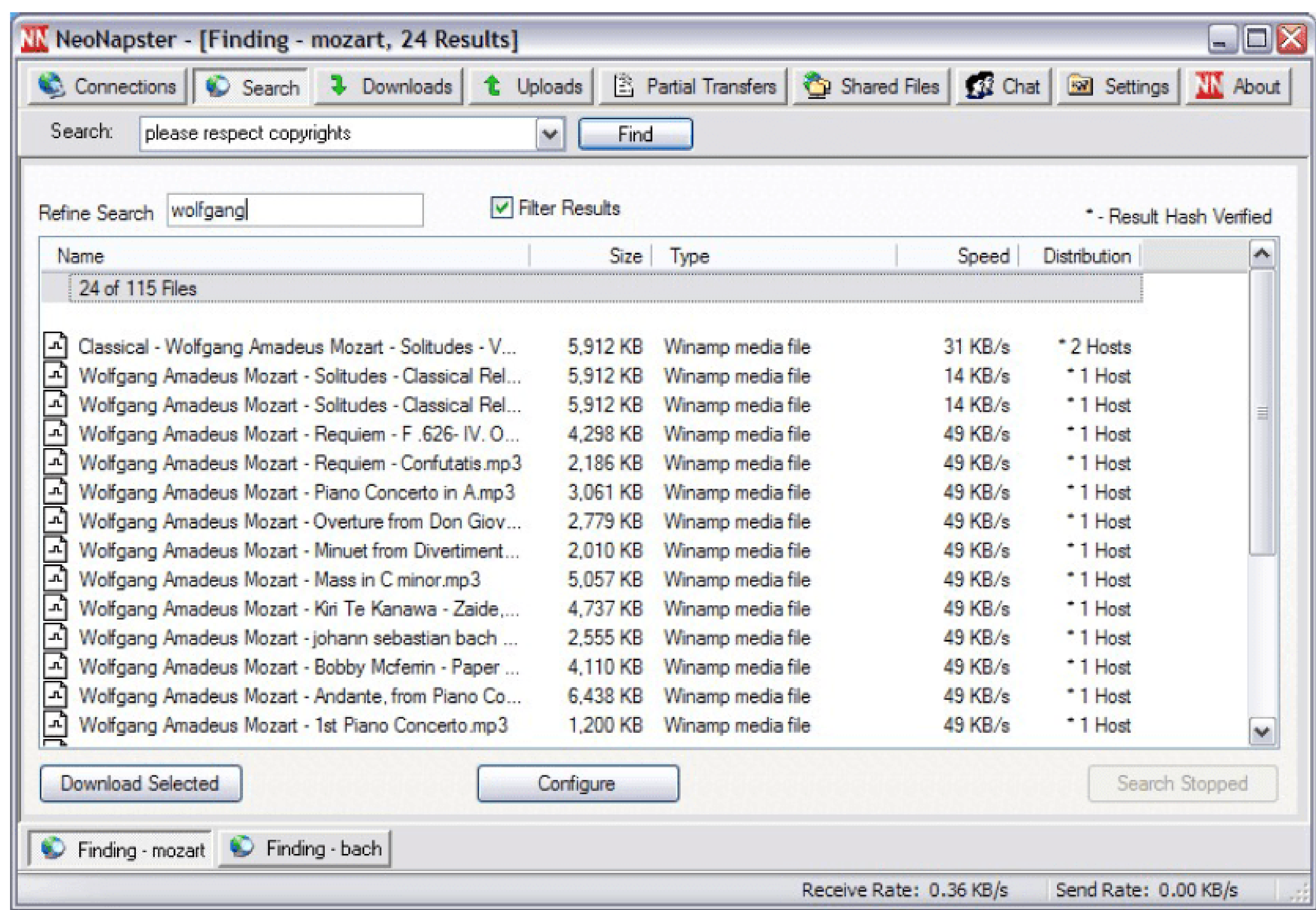


图 1.15　Napster 系统界面图

注意： 虽然 Napster 取得了巨大成功，但也带来了不良信息的传播，给版权保护、网络安全造成了严峻的挑战。最终，Napster 由于深陷版权问题的困扰，于 2001 年被迫关闭。

1.5　P2P 的发展实例

到了 2000 年，P2P 技术的发展显出了勃勃的生机，各种 P2P 应用系统和软件比比皆是，人们也在不知不觉中感受到了 P2P 作为高科技发展载体的快乐。下面就 P2P 技术在文件共享、资源下载、流媒体发布和即时通信等方面的发展作简要介绍。

1.5.1　P2P 文件共享系统——eMule

eMule 是一个开放的 P2P 文件共享系统，基于 eDonkey 的网络协议。eMule 这个名称来源于一个动物——骡，所以中文也称作电骡或骡子等。

在 2002 年 05 月 13 日，Hendrik Breitkreuz（亨德里克•布雷特克鲁兹）的 Merkur，他不满意当时的 eDonkey2000 客户端，并且坚信自己能做出更出色的 P2P 软件，于是便着手开发。他凝聚了一批原本在其他领域有出色发挥的程序员在他的周围，eMule 工程就此诞生。他的目标是将 eDonkey 的优点及精华保留下来，并加入新的功能以及使图形界面变得更好。

由于 eMule 是开放源代码，在许多优秀程序员的共同努力下，eMule 很快便成为 P2P 网络中文件共享的优秀系统之一，在开放源码编程中也一直处于最活跃的位置。图 1.16 是在开源社区（https://sourceforge.net）中一些著名的开源软件的下载排名情况，从图中可以看出，eMule 在开源社区的下载量是排在第一位的，足见其在网络中的位置。

SOURCEFORGE.NET®　Log in　Create account　Community　About　Help

Top Downloads • *For all time, updated daily*

View Other Top Categories

Rank	Project Name	Downloads
1	eMule	493,252,307
2	Azureus	386,589,815
3	Ares Galaxy	193,922,160
4	7-Zip	65,885,572
5	FileZilla	61,084,497
6	Audacity	58,986,756
7	GTK+ and The GIMP installers for Windows	58,465,573
8	DC++	55,431,577
9	BitTorrent	52,010,185
10	Shareaza	47,734,572
11	VirtualDub	45,496,382
12	PortableApps.com: Portable Software/USB	44,831,284
13	CDex	38,794,499
14	Pidgin	29,230,140
15	eMule Plus	28,519,226
16	guliverkli	28,136,447
17	aMSN	26,887,141

图 1.16　eMule 软件开源社区的下载排名情况

eMule 客户端使用多个途径搜索下载的资料源，ED2K、来源交换、Kad 共同组成一个可靠的网络结构。尤其是 eMule 的排队机制和上传积分系统有助于激励人们共享并上传给他人资源，以使自己更容易、更快速地下载自己想要的资源。

注意：关于 eMule 的详细知识，在本书的第 7 章会有详细的讲解。

eMule 的 Web 服务特性和 Web 服务器允许用户快速地从网络存取资料，能在下载时指定类别以组织和管理文件。eMule 还提供很复杂的搜索算法和一个大范围的搜索方式，使用户非常方便地找到想要的资源。使用 eMule 的信息及好友系统，能传送信息到其他客户端；使用内建的 IRC 客户端，能和全世界其他共享者聊天。在官方版本的基础上，有各种各样的修改版本（Mod），提供了各种不同的附加功能。并且这些 Mod 也都是开放源代码的，这些多方面的因素使得 eMule 的发展突飞猛进，成为了互联网上最热门的 P2P 应用之一。后面会对 eMule 系统及其涉及的各种技术进行详细地讲解。

如图 1.17 是 eMule 系统的主界面图，当前图中展示的是 eMule 在下载文件时的相关信息。eMule 作为一款 P2P 文件共享软件，是 P2P 技术的典型应用，也是资源共享方面的杰出代表，在后文中还会对 eMule 的各方面知识进行详细地讲解。

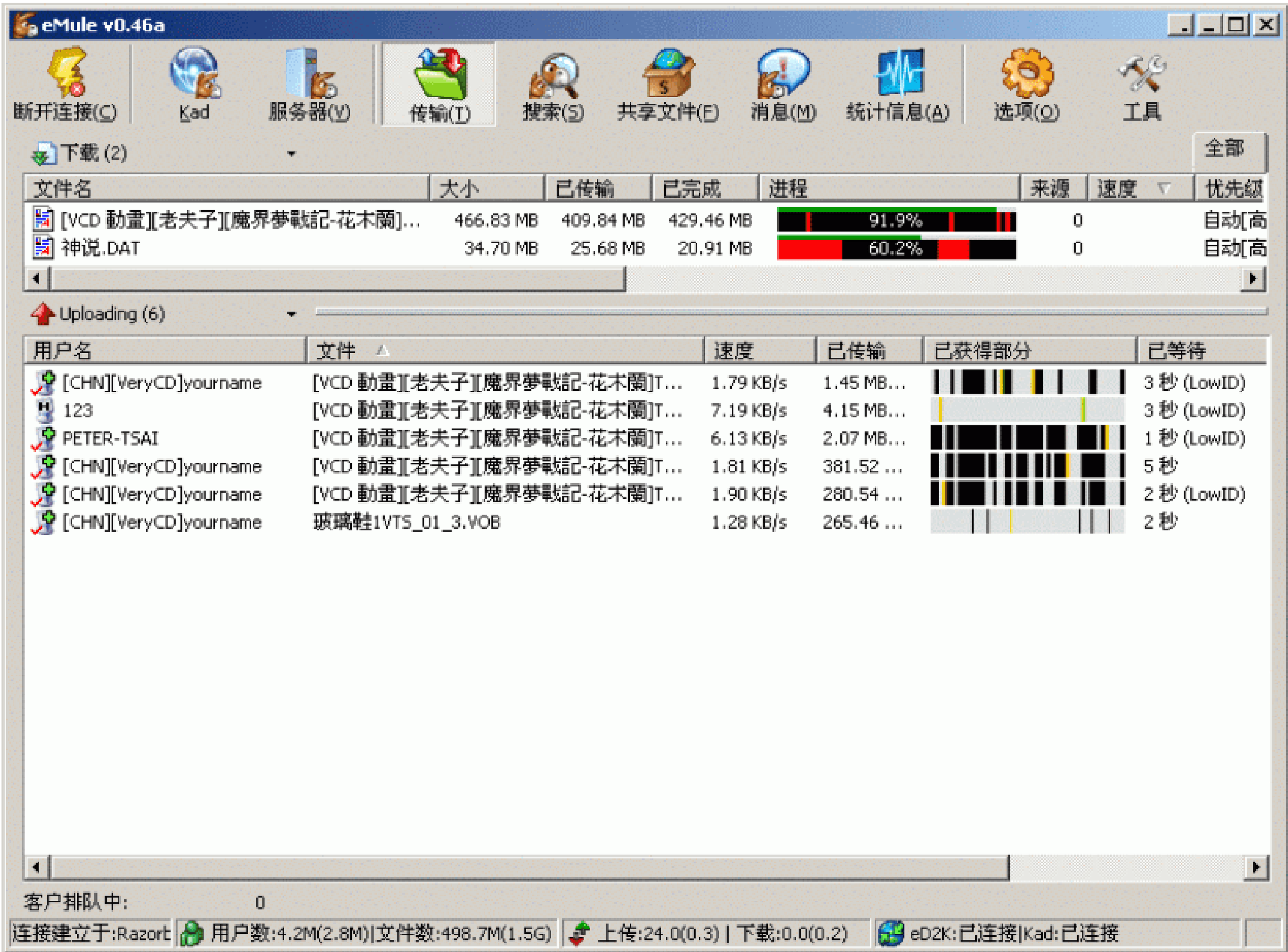


图 1.17　eMule 系统界面图

1.5.2　P2P 下载技术——迅雷（Thunder）

迅雷是一款智能下载软件，在网络中进行资源下载时它拥有比目前用户常用的下载软件快数倍的下载速度，这里提及迅雷，主要是因为它可以在 P2P 网络中搜索并下载资源。

迅雷是由深圳市迅雷网络技术有限公司发布的迅雷多媒体下载软件（迅雷官方网站为

www.xunlei.com），迅雷下载工具已经成为中国互联网最流行的应用下载软件之一。根据迅雷官方网站的介绍，迅雷下载软件使用的是一种多资源的超线程技术，基于网格原理，能够将网络上存在的服务器和计算机资源进行有效地整合，构成独特的迅雷网络，通过迅雷网络各种数据文件能够以最快的速度进行传递。迅雷的多资源超线程技术还具有互联网下载负载均衡功能，在不降低用户体验的前提下，迅雷网络可以对服务器资源进行均衡，有效降低了服务器负载，如图 1.18 是迅雷的系统界面图。

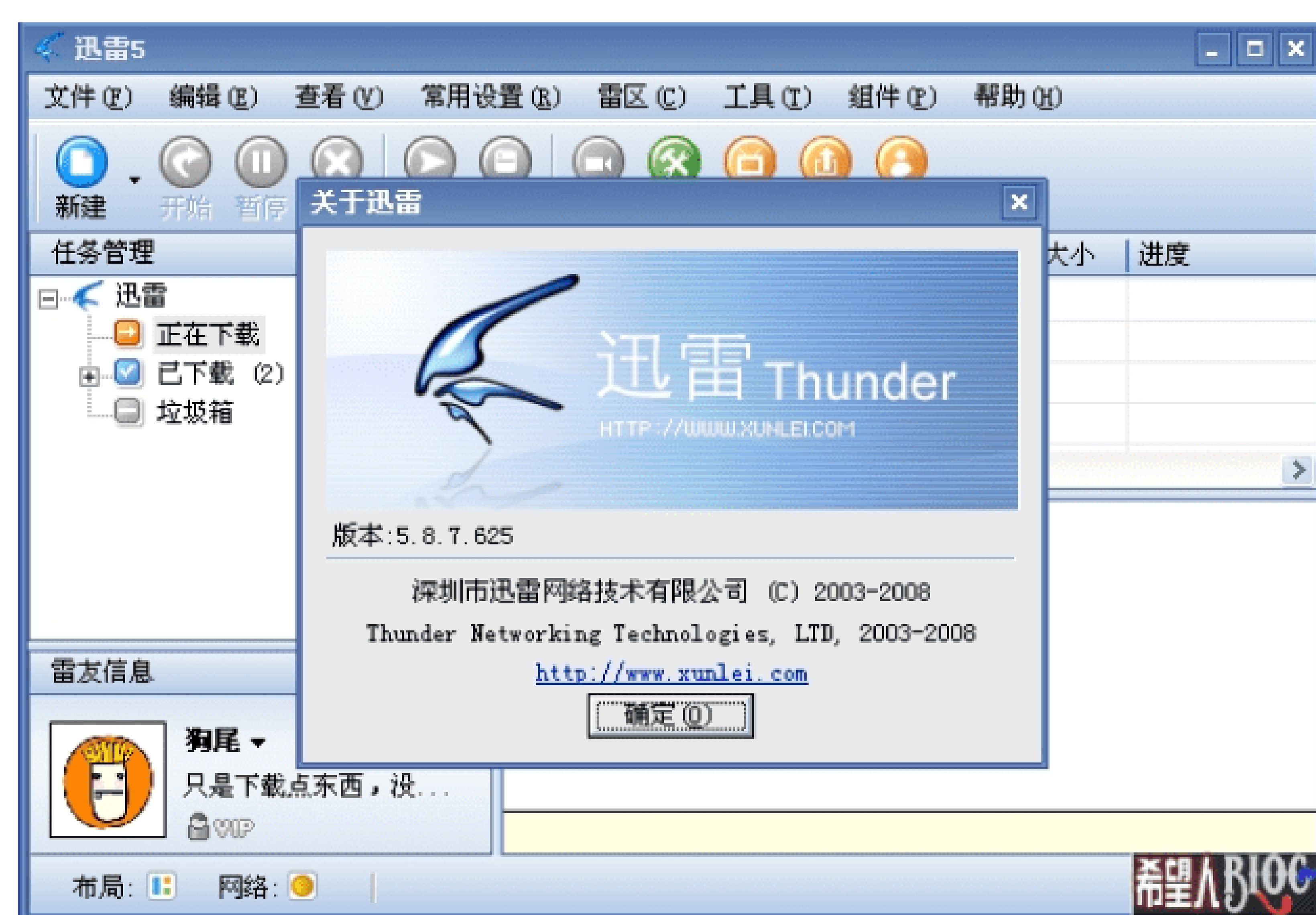


图 1.18　迅雷软件运行界面图

从技术的角度来分析，迅雷的技术主要分成两个部分。

- 一部分是对现有 Internet 下载资源的搜索和整合，将现有 Internet 上的下载资源进行校验，将相同校验值的统一资源定位（URL）信息进行聚合。当用户单击某个下载连接时，迅雷服务器按照一定的策略返回该 URL 信息所在聚合的子集，并将该用户的信息返回给迅雷服务器。
- 另一部分是迅雷客户端通过多资源多线程下载所需要的文件，提高下载速率。

迅雷高速稳定下载的根本原因，在于同时整合多个稳定服务器的资源以实现多资源多线程的数据传输。在使用迅雷进行下载的时候，每个用户在网上下载的文件都会在迅雷的服务器中进行数据记录，如有其他用户再下载同样的文件，迅雷的服务器会在它的数据库中搜索曾经下载过这些文件的用户，服务器再连接这些用户，通过用户已下载文件中的记录进行判断，如用户下载文件中仍存在此文件（文件如改名或改变保存位置则无效），用户将在不知不觉中扮演下载中间服务器的角色，在下载文件的同时也上传文件。

当前，迅雷已完可以完全支持 eMule 和 BT（BitTorrent，简称 BT，俗称比特洪流、BT 下载）下载，使用迅雷技术，可以在 P2P 网络中实现更快的资源下载。

1.5.3　P2P 文件传输技术——BT

BT 协议是一个网络文件传输协议，它能够实现点对点文件分享的功能。比起其他点对点的协议，它体现了更多 P2P 的特性，这个特性简单地说就是下载的人越多，速度越快。使用 BT 下载，用户下载完成后只要不停止任务，并继续上传所下载的文件，就可以成为

BT 网络中的一个种子，以服务器的角色把下载的文件分享出去让其他人下载。

普通的 HTTP/FTP 下载使用 TCP/IP 协议，BT 协议是架构于 TCP/IP 协议之上的一个 P2P 文件传输协议，处于 TCP/IP 结构的应用层。BT 协议本身也包含了很多具体的内容协议和扩展协议，并在不断扩充中，关于 BT 协议及下载原理如图 1.19 所示。

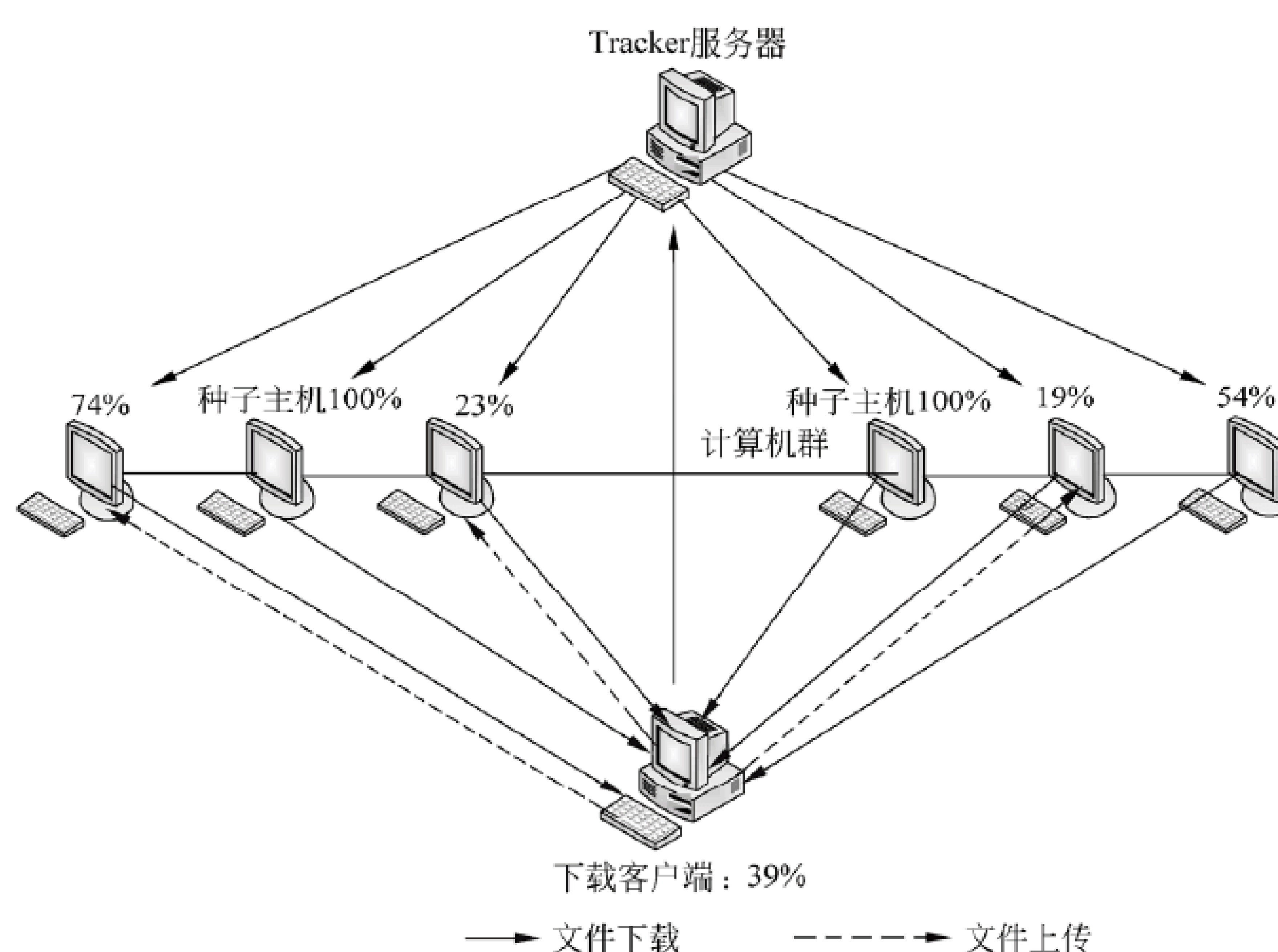


图 1.19　BT 协议及下载原理图

根据 BT 协议规范，文件发布者会根据要发布的文件信息生成一个包含元文件信息的.torrent 文件，即种子文件，也简称为“种子”。.torrent 文件本质上是文本文件，包含 Tracker 信息和文件信息两部分。Tracker 信息主要是 BT 下载中需要用到的 Tracker 服务器的地址和针对 Tracker 服务器的设置信息，图 1.19 中，处于最上面位置的就是 BT 网络中的 Tracker 服务器。

下载者，也就是参与到 P2P 网络中下载资源的一个客户端主机，在 BT 网络中称为 Peer 结点，图 1.19 中，位于最下端位置的就是 BT 网络中一个普通的 Peer。当这个 Peer 要下载文件内容时，需要先得到相应的.torrent 文件，然后使用 BT 客户端软件进行下载。下载时，BT 客户端首先解析.torrent 文件得到 Tracker 服务器地址，然后连接 Tracker 服务器。Tracker 服务器回应下载者的请求，并提供给下载者其他拥有此资源的 Peer（资源的发布者）的 IP 地址和端口，图 1.19 中，位于中间一排的 PC 都是 Tracker 服务器检测到的拥有此资源的 Peer。

得到这些 Peer 信息后，下载者根据 IP 地址、端口等与其他 Peer 进行连接，通过解析.torrent 文件，双方进行交流对比目前已有的数据块，然后将对方所没有的数据进行交换。在资源的交互过程中，不需要其他服务器参与，这样两个 Peer 结点之间就可以进行对等的点到点资源交互和传输了。图 1.20 是 BT 下载客户端 BitComent 的系统界面图。

注意：BT 中，BT 种子文件信息是根据对目标文件的计算生成的，计算结果根据 BT 协议内的 B 编码规则进行编码。它的主要原理是需要把提供下载的源文件虚拟分成大小相等的块，块大小必须为 2k 的整数次方，并把每个块的索引信息和 Hash 验

证码写入.torrent 文件中，所以.torrent 文件简单地说就是被下载的源文件的“索引”信息。

一般的 HTTP/FTP 下载，发布文件仅在某个或某几个服务器，下载的人太多，服务器的带宽容易不胜负荷，变得很慢。而 BT 文件传输技术协议下载的特点是，下载的人越多，提供的下载源也就越多，种子也会越来越多，下载速度就越快。通过这种方式，分散了单个线路上的数据流量，也减轻了服务器负担。BT 文件传输技术是 P2P 技术的又一经典应用，最大限度地利用了网络资源和带宽，实践了“人人为我，我为人人”的网络传输目标。

如图 1.20 是 BT 系统的主界面，关于 BT 的详细技术，在后面也会讲到，这里只是对基本知识作简要说明。

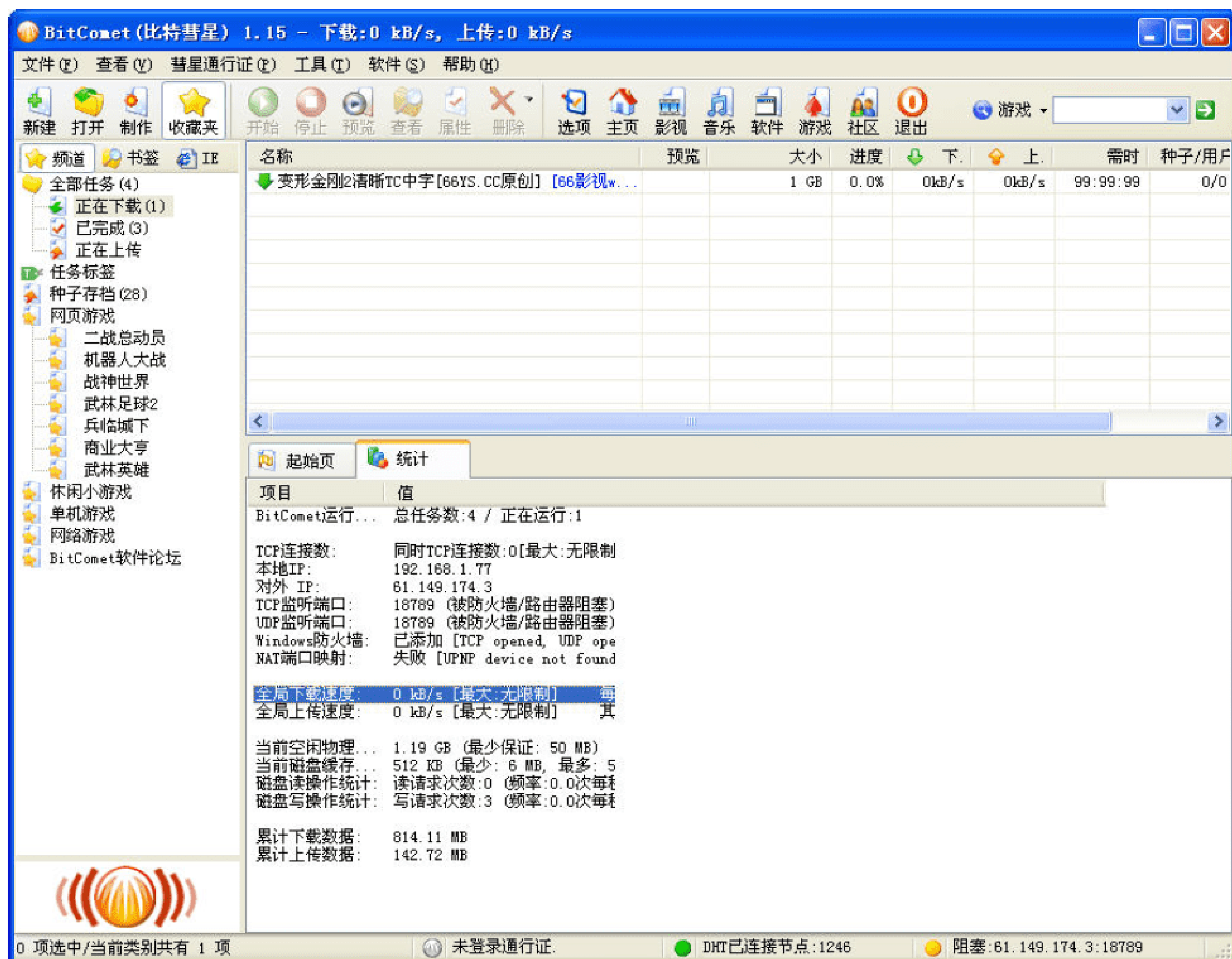


图 1.20　BT 系统主界面

1.5.4　P2P 视频分发技术——PPLive

PPLive 是一款用于互联网上大规模视频直播的免费软件，是基于 P2P 技术的流媒体发布和传输方面的典型代表，也是 P2P 技术在视频直播、点播应用上的成功范例。PPLive 内核采用了独特的 ALM 多播和内聚算法技术，运用 P2P 的下载和传输原理，有效地降低了视频传输对运营商主干网的冲击，减少了出口带宽流量，并能够实现用户越多播放越流畅的特性，使得整体服务质量大大提高。PPLive 作为一个视频发布系统，在进入商业运作后，

它以 P2P 网络电视软件（PPLive 视频系统）和视频节目信息发布网站（www.PPLive.com）为主要平台载体。在同类网络电视直播平台中，PPLive 的用户数量、版权资源、直播频道、节目质量和日访问量等都位于前列，目前在国内知名度较高。

PPLive 与传统的网络电视相比，其成功的一条重要原理就在于它充分利用了 P2P 原理的巨大技术优势。在网民心目中，以往想在网络上看电视，只能登录提供网络电视服务的网站，然后通过某一网络播放器来播放在线视频，随着在线用户的增多，服务器就会超负荷运转，画面流畅性和声音的质量都会变得非常差。而 PPLive 是基于 P2P 原理的，将传统的 S/C 模式变成 P2P 的模式，用户越多，速度反而越快，彻底改变了用户量和网络带宽之间的矛盾，在市场竞争中自然就占据了有利地位。如图 1.21 是 PPLive 系统的主界面图。



图 1.21　PPLive 系统至界面

1.5.5　P2P 网络语言通信技术——Skype

Skype 是基于 P2P 技术的网络语音沟通工具。它可以提供免费高清晰的语音对话，也可以用来拨打国内国际长途，还具备即时通信所需的其他功能，比如文件传输、文字聊天等。

Skype 是在 KaZaA 的基础上开发的，就像 KaZaA 一样，Skype 本身也是基于覆盖层的 P2P 网络。在 Skype 的网络结构中，网络中有两种类型的结点，即普通结点和超级结点。普通结点是能传输语音和消息的一个功能实体；超级结点则类似于普通结点的网络网关，所有的普通结点必须与超级结点连接，并向 Skype 的登录服务器注册自己的信息加入到 Skype 网络中。Skype 的登录服务器上存有用户名和密码，并且授权特定的用户加入 Skype 网络，如图 1.22 展示了 Skype 系统的主界面。

Skype 的另一个突出特点就是能够穿越地址转换设备和防火墙。Skype 能够在最小传

输带宽 32kb/s 的网络上提供高质量的语音。Skype 是 P2P 网络即时通信技术的代表。由于其具有超清晰语音质量、极强的穿透防火墙能力、免费多方通话以及高保密性等优点，成为互联网上即时通信方面使用最多的 P2P 应用之一。

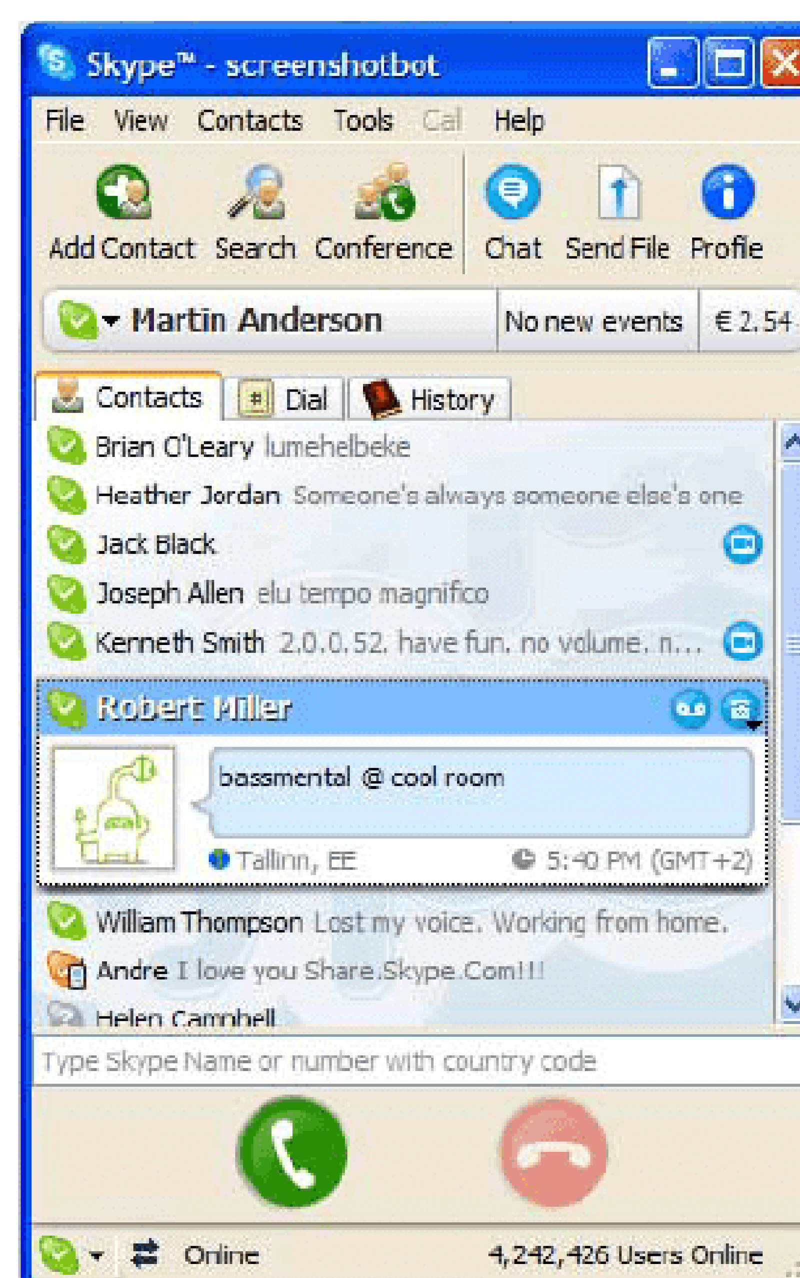


图 1.22　Skype 系统的主界面

注意： KaZaA 是一款非常优秀的基于 P2P 的点对点文件共享工具，提供会员们下载分享出来的资源，通过简易的搜寻，可以快速地找到想要的资源，包括音乐、影片、软件、游戏、图片以及文件等。

当前在互联网中基于 P2P 的协议和应用远远不止以上说的这些，在 P2P 技术快速发展的今天，各类与 P2P 相关的技术、协议、应用系统数不胜数，这里就不再一一介绍。在 wikipedia 上有对当前互联网中用到的 P2P 网络和协议的详细总结，读者可自行访问网站 http://en.wikipedia.org/wiki/Peer-to-peer，以参考相关的信息。

1.6　P2P 的技术特点及存在的问题

通过以上讲述可以看出，与其他的网络模型相比，P2P 网络有着自身的特点，正是因为这些特点促进了 P2P 技术的快速发展，而 P2P 技术在发展的过程中也同样面临着诸多的问题，本节重点讲述 P2P 的主要特点及存在的主要问题。

1.6.1　P2P 网络的特点

P2P 网络作为架构在 Internet 之上的虚拟网络，除了拥有一般网络的特点以外，它还有自身的其他特点，主要有以下几个方面。

1．P2P的网络虚拟性

P2P 网络并不是一个独立的网络概念，是一种构筑于因特网基础设施之上的虚拟重叠网络。P2P 网络是由对等点构成，网络中不存在中心服务器，没有固定的路由设备，网络的计算资源、存储资源、内容资源均来自于 Internet 中的参与 P2P 系统的各对等点。这也为提高网络扩展性、降低网络服务成本提供了可能性。

2．P2P的拓扑动态性

P2P 网络是一种分布式的动态网络，随着各对等点的动态加入与退出，整个网络的资源总量、拓扑结构、路由方式始终处于动态变化中。P2P 网络会随着结点的增加其资源总量和服务能力也会同步地扩充，网络的动态性始终能较容易地满足用户的需要。整个体系

是动态的、全拓扑分布的，不存在瓶颈。理论上其动态扩展性几乎可以认为是无限的。

3．P2P的网络健壮性

P2P 架构天生具有耐攻击、高容错的优点。由于服务是分散在各个结点之间进行的，部分结点或网络遭到破坏对其他部分的影响很小。P2P 网络一般在部分结点失效时能够自动调整整体的拓扑结构，保持其他结点的连通性。P2P 网络通常都是以自组织的方式建立起来的，并允许结点自由地加入和离开。P2P 网络还能够根据网络带宽、结点数、负载等变化不断地做自适应式的调整。

4．P2P的负载均衡

P2P 网络环境下由于每个结点既是服务器又是客户机，减少了对传统 S/C 结构模式下服务器计算能力、存储能力的要求，同时因为资源分布在多个结点中，更好地实现了整个网络的负载均衡。

5．P2P在应用中的高性价比

性能优势是 P2P 被广泛关注的一个重要原因。随着硬件技术的发展，个人计算机的计算和存储能力以及网络带宽等性能依照摩尔定理高速增长。采用 P2P 架构可以有效地利用互联网中散布的大量普通结点，将计算任务或存储资料分布到所有结点上。利用其中闲置的计算能力或存储空间，达到高性能计算和海量存储的目的。通过利用网络中的大量空闲资源，可以用更低的成本提供更高的计算和存储能力。

6．P2P是处于应用层的网络

P2P 网络是面向业务的网络，是一个应用层的网络。P2P 网络业务范围广泛分布于文件共享、数据存储共享、分布式计算、网络游戏、IP 软件电话、流媒体分发、多媒体通信等应用领域。

1.6.2　P2P 发展带来的问题

在说尽 P2P 的好时，也不得不想到，就如同历史总是在曲折中前进，任何新事物的发展总不会是一帆风顺的。P2P 也一样，在 P2P 辉煌发展的同时也面临着诸多的问题。

1999 年 5 月，由范宁和帕克共同创办的文件共享社区网站——Napster 正式成立，在辉煌一时之后，却由于版权问题，最后被迫关闭，这时 P2P 首先需面对一个现实，在充分享受自由与共享的 P2P 网络世界里如何保护知识产权？同时，还有一系列其他的问题，总结一下，基本有以下几个大的方面。

1．知识产权问题

P2P 网络是一个高度自由、自治、共享的网络，其网络结构中没有一个统一管理的机制来对网络中的各个用户进行认证和授权管理。P2P 网络中分布的资源是分散的、无约束的，这在 P2P 共享的网络机制下自然就加速了盗版资源的分发，侵犯了版权，也增加了知识产权保护的难度。

2．黑客和病毒入侵问题

在 P2P 网络中，资源都来源于各个分散的结点中，在使用 P2P 进行资源的传输与交互时，要验证共享资源的来源是否安全是非常困难的，这就不可避免地给黑客和病毒入侵带来机会，给网络安全带来隐患，P2P 网络常被攻击者作为传递恶意代码的载体。由于在 P2P 网络中，每个结点防御黑客和病毒攻击能力是不同的，因此只要有一个结点遭到攻击，就可以通过内部共享和通信机制将攻击扩散到附近的邻居结点，在短时间内可以造成网络拥塞甚至瘫痪。因此，黑客和病毒的潜在危机对 P2P 系统安全性和健壮性提出了更高的要求，也是目前 P2P 技术发展必须要解决的问题。

3．敏感信息的泄漏问题

当使用 P2P 传输信息时，P2P 软件几乎都设定使用者只能传输特定数据中的某些信息的能力，但如果遭到恶意代码的攻击，就可能门户大开，在不知不觉中会给其他用户访问他人敏感信息的访问权限。这样可能造成有意图的非法访问，以入侵给个人、公司或国家的敏感信息。如获取用户的电脑系统资料、个人信息、银行卡的用户名和密码、公司客户往来信息、国家机密等，因此，在 P2P 网络中，需要考虑如何保护用户的安全策略。

4．带宽问题

由于许多 P2P 应用程序同时运行客户端和服务器端的应用程序，所以它对上行带宽和下行带宽都要占用。而且很多 P2P 应用使用了泛洪式（Flooding）的查询技术，这就会导致大量的广播消息充斥整个网络，增加网络流量。这对大多数提供非对称带宽通道的 ISP 来说，如果不进行专门的配置就很难适应 P2P 程序的运行。

由于 P2P 软件方便快捷的特性，使得更多的人愿意从中获取想要下载的文件，再加之此类软件多以音频、视频为主，所以 P2P 占领了大量的网络带宽。例如，从 1997 年 10 月到 2006 年底，中国运营商的长途骨干网络中超过一半的流量是基于 P2P 应用的，多数地区和省份的流量达到 60%～70%，有的高达 90%。P2P 软件运行，严重时可能会造成网络阻塞，影响网络正常运行。

说明： P2P 存在的问题正是 P2P 发展的动力，P2P 也正是在这种交织着曲折与辉煌的道路上慢慢成长起来的。

1.7　P2P 的研发与未来

在 P2P 技术及其应用系统在产业界迅速发展的同时，学术界、研究界也及时跟进，对 P2P 系统展开了大规模的研究工作。自 2000 年起，在 OSDI、SOSP、SIGCOMM、USENIX、HOTOS 等系统结构方向的顶级会议上不断出现关于 P2P 系统的重要研究成果。

2001 年，学界又召开了新的专门针对 P2P 系统的学术会议 IPTPS，由于该会议受到各著名院校和研究机构的广泛关注，很快成为 P2P 研究领域的高峰会议，发表了大批优秀论文，成为 P2P 研究的风向标。从 2002 年开始，Berkeley、Stanford 等著名大学相继开设了

P2P 相关的研究生课程，进一步推广了 P2P 这一新兴技术的研究，没有研发就没有 P2P 的未来，本节就介绍一下当前 P2P 的研发概况及其未来的发展。

1.7.1　国内学术机构研发情况

针对 P2P 技术的研究，在国内也发展的十分迅速。从商业的角度看，一大批有着自主知识产权的 P2P 应用软件相继诞生，尽管它们的出现是以盈利为目的，但无形中也带动了 P2P 技术的发展。而从技术的角度对 P2P 技术展开研究的，主要集中在一些高校和学术机构，比较典型的有以下几个。

1．Maze

Maze 是北京大学网络实验室开发的，一个中心控制与对等连接相融合的对等计算文件共享系统，在结构上类似 Napster，对等计算搜索方法类似于 Gnutella。网络上的一台计算机，不论是在内网还是外网，可以通过安装运行 Maze 的客户端软件自由加入和退出 Maze 系统。每个结点可以将自己的一个或多个目录下的文件共享给系统的其他成员，也可以分享其他成员的资源。Maze 支持基于关键字的资源检索，也可以通过好友关系直接获得，网址为 http://maze.tianwang.com/。

2．Granary

Granary 是清华大学自主开发的 P2P 存储服务系统，其设计目标是能够自适应地支持高动态系统和稳定系统，并提供面向对象的存储服务。Granary 采用清华大学提出的 Tourist 路由算法，这种算法可以根据网络的动态情况自适应地改变自己的路由表大小，使得网络在稳定环境中获得 O(1)复杂度的路由效率，而在动态环境中至少保证 O(logn)复杂度的路由效率。

Granary 采用完全副本的数据冗余方式，冗余后的数据被 Granary 均匀地分发到网络中的结点上，以防止相关性错误，冗余数据所在结点的 IP 列表以及数据对象的属性被作为元数据，以 DHT 的方式存储在 P2P 网络中。Granary 利用 PeerWindow 算法向目录结点广播结点加入和离开的事件，系统为用户存放 2 倍于用户指定的数据副本数，当系统中副本数低于用户指定的副本数时，Granary 才触发修复操作。

利用路由算法的自适应性，Granary 可以自适应不同动态性的环境以向互联网用户提供一种高效的存储服务。

注意：所谓 P2P 存储服务系统，是指存储服务的提供者在 Internet 中部署一定数量的存储服务器，为用户提供数据存储服务，确保数据的可靠性、可用性、安全性和访问效率。存储服务的使用者按照所存储数据的容量和质量付费。

3．AnySee

AnySee 是华中科大设计研发的视频直播系统。它采用了一对多的服务模式，支持部分 NAT 和防火墙的穿越，提高了视频直播系统的可扩展性；同时，它利用近播原则、分域调度的思想，使用 Landmark 路标算法直接建树的方式构建应用层上的组播树，克服了 ESM

等一对多模式系统由联接图的构造和维护带来的负载影响。

4．WonGoo

WonGoo 是中科院计算所研制的一套 P2P 技术平台，该平台主要为信息安全、网格计算提供支撑技术和试验环境。同时 WonGoo 的基础部件将在开发完善之后以开放源代码的方式向社会公开。

WonGoo 主要包括两个方面的特色功能，即具有强匿名性的 P2P 通信（WonGoo-Link）及基于内容查找的 P2P 资源共享（WonGoo-Search），可以在这两个功能的基础上搭建各种特色化的 P2P 应用，目前相关的应用还没有具体实现。WonGoo-Link 与 WonGoo-Search 可以分别独立构造并搭建各自的应用。同时，WonGoo-Search 底层通信也可以采用 WonGoo-Link 协议来实现更安全的应用。

5．基于IPv6的P2P内容存取应用系统

这是北京大学、清华大学、上海交通大学、浙江大学、华中科技大学、华南理工大学、北京世纪鼎点软件有限公司共同承担的国家 CNGI 项目的一部分。它主要研究基于智能结点弹性重叠网络技术的内容存取应用中间件系统，在 CNGI 上建设可管理、可控制和可运营的智能结点弹性重叠网络，开发内容存取类应用。

1.7.2　国内企业研发的情况

国内企业在 P2P 的应用领域研究一直与世界同步，开发了众多使用广泛的 P2P 产品。这些产品主要集中在文件共享与下载，网络流媒体电视等方面。

1．POCO

POCO 是中国领先的免费电影、音乐、动漫等多媒体分享平台，同时在线人数突破数 70 万人，是中国最大的电影音乐动漫分享平台。它是有流量控制力的、无中心服务器的第三代 P2P 资源交换平台。POCO 提供多点传输、断点续传等技术，来保障传输过程的高效和稳定。

2．OP

OP 又称为 Openext Media Desktop，是一个网络娱乐内容平台，Napster 的后继者。它可以最直接的方式找到你想要的音乐、影视、软件、游戏、图片、书籍以及各种文档，随时在线共享文件容量数以亿计“十万影视、百万音乐、千万图片”。OP 整合了 Internet Explorer、Windows Media Player、RealOne Player 和 ACDSee，是国内的网络娱乐内容平台。

3．PPLive

PPLive 是一款用于互联网上大规模视频直播的共享软件。它使用网状模型，有效解决了当前网络视频点播服务的带宽和负载有限问题，实现了用户越多播放越流畅的特性，整体服务质量大大提高。

4．PPStream

PPStream 是一套完整的基于 P2P 技术的流媒体超大规模应用解决方案，包括流媒体编码、发布、广播、播放和超大规模用户直播。能够为宽带用户提供稳定和流畅的视频直播节目。与传统的流媒体相比，PPStream 采用了 P2P-Streaming 技术，具有用户越多播放越稳定，支持数万人同时在线的大规模访问等特点。

其他的非常优秀的商业软件包括 PP 点点通，eMule、BT 客户端软件等，具体详情读者可以访问中国 P2P 门户网站 http://www.ppcn.net/了解详细信息，这里不再赘述。

注意：以上资源部分来源于中科院计算技术研究所，罗杰文同志的“Peer-To-Peer 综述”，详情可参阅：http://www.intsci.ac.cn/users/luojw/P2P/ch01.html。

1.7.3　P2P 的新机遇

P2P 的特性以及其展示出来的强大生命力，让这一技术在当前及可遇见的将来都面临着诱人的发展机遇，这些机遇主要体现在它巨大的商业价值以及对互联网秩序的变革上。

1．P2P的商业价值

与旧有模式比较，P2P 为整个网络概念及网络运营商都带来了很多新的机会。比如采用 P2PCDN 技术可以将 P2P 流量尽量限制在局部范围内，减少跨运营商网络的流量以减少结算费用。合理制定网际互联网结算规则是政府进行调控的重要手段。

P2P 文件共享下载可以用于文件的合法分发和传播方面，有利于发挥互联网无所不在的优势。P2P 的流媒体分发技术，可以有效突破带宽的瓶颈，高效地分发声音、视频、图像等多媒体信息，营造一个丰富多彩的互联网世界。P2P 在协同计算、即时通信方面也都有杰出的表现，其潜在的特性和对网络应用需求的吻合，使 P2P 拥有巨大的商业价值。

2．建立网络新秩序

P2P 是一种架构在互联网之上的虚拟重叠网站。目前可管理的 IP 网和未来的 NGN 采用集中的管理模式，不能适应管理 P2P 的需求，需要发展新的分布式的管理系统。目前，计算机网络界正在研究开发智能结点重叠网分布管理系统，另外，借助于 IPv6 技术的发展，未来以 NAT 技术来共享上网从而隐藏了使用者真实身份的情况将不复存在，IPv6 将有可能为 P2P 网络提供一种实名制的应用环境。一旦这些技术实现突破并得以运用，整个网络的管理概念、使用机制、管理秩序都会发生新的改变。

1.7.4　P2P 的应用简述

与传统的分布式系统相比，P2P 技术具有无可比拟的优势。同时，P2P 技术具有广阔的应用前景。根据具体应用不同，可以把 P2P 分为以下这些类型。

1．资源共享

提供文件和其他内容共享的 P2P 网络，例如 Napster、Gnutella、eDonkey、eMule、BT 等。

在 P2P 文件共享领域，技术已经比较成熟，这样的软件如 eMule、BT、KaZaa、POCO 等，它们都分别培养了自己的用户群。但是，由于基于不同协议的 P2P 系统资源并不共享相互隔绝，所以当前阶段，这一类型软件正处在自由竞争阶段，进入市场的企业无论是规模还是实力，都不相上下。要想在竞争中取胜，下面这些内容必须考虑。怎样激励用户提供资源？怎样保障网络里资源高速稳定的下载速度？怎样去除间谍软件和病毒在系统中的传播？除了这些，还有人气的较量、服务质量的较量、收费与免费的较量等。P2P 技术在资源共享方面，最终研究方向是 P2P 网间资源的整合，资源互通，搜索共享。

2．协同计算

P2P 在对等计算能力和存储共享能力方面也有突出的表现，例如 SETI@home、Avaki、Popular Power 等；基于 P2P 方式的协同处理与服务共享平台，例如 JXTA、Magi、Groove、.NET My Service 等。

在 P2P 协同计算方面，国内企业起步较晚，相关产品还不是很多，而国外例如 Groove 在这方面已经作了大量的工作，开发了相对成熟的产品。随着协同计算概念的兴起，这方面软件的需求呈现急剧增长的趋势，应该具有广阔的应用前景。而且，这类软件往往是面向企业和政府用户，所以相对于免费的 P2P 文件共享软件来说，有更好的盈利空间。

3．流媒体分发

在 P2P 的流媒体技术方面，虽然已经有很多成熟的系统，但还有许多实质性的问题需要解决。由于 P2P 流媒体系统中结点的行为具有 Ad-Hoc 性质，如何在动态的系统环境下保证流媒体的服务质量，需要结合流媒体对 QoS 的要求和网络流量分析等方面的知识，并研究高效率、低代价的 QoS 保障机制。可研究的方向包括服务结点的选择、结点失效时如何保证流媒体服务的连续以及对多个发送端的传输调度等。

注意： Ad Hoc 网络是一个没有有线基础设施支持的移动网络。在 Ad Hoc 网络中，所有的结点都是由移动主机构成的。网络拓扑结构的动态性是 Ad Hoc 网络的重要特点。Ad Hoc 网络通信的核心问题在网络通信效率和结点能量消耗之间的合理平衡。而 QoS 的英文全称为 Quality of Service，中文名为服务质量。QoS 是网络的一种安全机制，是用来解决网络延迟和阻塞等问题的一种技术。

4．P2P即时通信

P2P 的即时通信交流功能是在人们日常的互联网生活中应用最多的功能之一，最常见的一些即时通信软件如 QQ、ICQ、OICQ、Yahoo Messenger 等，几乎每天都会使用。P2P 的即时通信还包括安全的 P2P 通信与信息共享，典型的应用代表就是 Skype，其他的还有

Crowds、Onion Routing 等。

1.7.5　P2P 在中国的发展

P2P 技术为互联网的发展带来了深远的影响，也为 IT 产业带来了无限的商机。对于 P2P 未来发展趋势的探讨和研究，一直是业界和学界关注的焦点之一。

P2P 在中国的发展并非一帆风顺，它经历了繁荣，也遭遇过停滞，目前正在迎接新一轮的发展。P2P 在中国能否再创辉煌？谁将在新一轮的竞争中成为赢家？中国 P2P 的发展将会何去何从？

互联网领域的专业研究咨询机构——互联网实验室曾推出了《P2P 的现状及发展趋势研究报告》，以丰富的资料研究和深入的案例分析，在立足 P2P 发展现状的同时，对国内外 P2P 的发展趋势做了全面客观的预测，对 P2P 企业所关心的一切关于 P2P 现状和发展趋势的问题，都给予了专业解答。

从目前的状况来分析，中国的 P2P 市场正处在自由竞争阶段，进入市场的企业无论是规模还是实力，都不相上下。因此，在较长的一段时期，中国的 P2P 市场将延续群雄逐鹿的竞争局面。

当前 P2P 行业标准尚未形成，而与 P2P 有关的相关应用又及其广泛，技术门槛与经营门槛都相对较低，是这种竞争局面长期存在的主要原因。在尝试了 P2P 在即时通信、音乐共享与下载等方面的产品开发与经营后，P2P 在中国的发展将会以探求 P2P 更多的商业应用为核心。在这方面，国外许多 P2P 企业就走在了前面。无论是协同办公，还是分布式计算，国外的 P2P 企业都有过比较成功的探索，为国内 P2P 企业提供了许多值得借鉴的经验。而国内立足于这两方面应用的 P2P 企业，数量很少，规模也远不及国外的竞争对手。事实上，面向企业的 P2P 应用，在中国市场是大有可为的。

在中国，电子商务的市场发展也是潜力无穷的，在互联网实验室新近推出的《P2P 的现状及发展趋势研究报告》中有数据显示：到 2003 年底，中国的电子商务市场 B2B 和 B2C 的交易总额将可能达到 40 亿美元之巨，B2B 的年均增速为 19.4%，而 B2C 的年均增速是 27.4%，均呈倍数增长。这样广阔的发展前景，势必为 P2P 在电子商务领域的应用带来无限商机。

P2P 对于用户最大的意义，不是它的技术和功能，而是它的理念。这种理念源于人们对互联网的憧憬和梦想，它使网络回归到 Internet 的本质，让共享与自由的精神充满网络世界。中国 P2P 的发展，必将经历一个从技术到理念的过渡，技术的不断进步为中国 P2P 的发展铺平道路，而理念的不断更新，则为中国 P2P 的发展指明方向。未来的竞争，不再仅仅以技术为导向，谁能以 P2P 的理念创造为网民所接受和留恋的产品和文化？谁才会最终夺得 P2P 胜利之杯？

注意：以上资料来源于互联网实验室，详情请参阅 http://www.chinalabs.com/。

2007 年 12 月 2 日，互联网协会秘书长黄澄清在“中国 IT 两会”上表示，中国已成为 P2P 流媒体发展最快的国家。截止 2007 年上半年，中国网民数量增长率为 31.7%。从网络视频市场观察，中国已成为 P2P 流媒体发展最快的国家。与之相伴的网络视频搜索也迅速发展，正成为互联网另一个热点。总体来看，中国的 P2P 技术及其产业，发展态势良好，

发展前景广阔，正以稳健、积极的步伐向前迈进。

1.8 本 章 小 结

P2P 应用回归互联网对等连接的本性，充分发挥互联网无所不在的优势，有着广阔的发展前景。尽管 P2P 应用还存在管理、安全、运营模式、知识产权和政策法规等问题，但是它对未来网络的影响是不容置疑的。而这些问题也是目前正在研究解决的热点问题，蕴藏着巨大商机。

综上所述，P2P 技术正处在发展的春天，基于这项技术的“杀手级”应用将不断涌现，这些技术将极大改善整个 IT 世界的面貌，可以说是互联网技术的又一次新的革命。

第 2 章　P2P 网络拓扑结构

P2P 网络，其本身是一个广泛而复杂的概念，在现实世界中，P2P 网络并不是作为一个独立的个体网络而存在的，无须人为的部署，也不需要专门的硬件组网，它是依赖于某一 P2P 应用系统而自发组成的网络。从 Internet 的角度来讲，P2P 网络是叠加在底层通信网络基础设施之上的重叠网络，是网络中的网络，是一个分布式的、具有互操作性的自组织系统。研究 P2P 网络的拓扑结构，对理解 P2P 网络的特点、P2P 网络的搜索机制、数据传输机制、结点发现以及充分利用并发挥 P2P 的功能都有十分重要的作用。

本章将重点学习 P2P 网络的拓扑技术，理解 P2P 不同的网络分类及组网特点，掌握 P2P 网络的结构及其应用原理。本章主要讲解的知识点如下。

- 互联网的整体结构模型：整体上了解整个 Internet 的结构及互联的特点。
- 网络拓扑：理解网络拓扑的概念，不同的拓扑分类及特点。
- P2P 的网络分类：掌握当前 P2P 网络的分类标准和主要的类别。
- 集中式的 P2P 网络：集中式 P2P 网络的概念、特点、原理，集中式 P2P 网络的典型应用、存在的问题等。
- 分布式 P2P 网络：分布式结构化 P2P 网络和分布式非结构化 P2P 网络的各自特点、原理，以及这类 P2P 系统的典型应用。
- 混合式 P2P 网络系统：集中式 P2P 和分布式 P2P 的有效综合，取两者的优点形成的较理想的 P2P 网络模型。
- 发展中的 P2P 网络系统：是一个形成中的概念，也是 P2P 网络结构重点研究的领域。

2.1　谜一样的网络世界

每天我们都会上网，都会通过网络与人交流思想、分享资源。通过网络，人们可以工作、购物、贸易、上课、休闲娱乐……几乎无所不能。在这个看得见的网络天地里，或许就是一台电脑、一根网线、一个浏览器而已，而那些我们看不见的则是一个如谜一样的网络世界。

注意：P2P 网络被称为是网络中的网络，是一个虚拟的重叠网络，要学习 P2P 的网络结构就需要对互联网络的结构、拓扑有所了解，本节是学习 P2P 网络拓扑的前导知识。

2.1.1　走进网络世界

在这个世界上有一个全球连接的网络吗？即你相信世界上的人与人之间有某些看不

见的通道相连接吗？你相信你的起心动念会影响到的不只是周遭的人，甚至远在地球另外一边的人都会受影响吗？也许有的人会说这正是混沌理论中的蝴蝶效应，北京的一只蝴蝶拍动翅膀，却引起美洲沙漠上的大风暴，的确，在网络世界里，这个效应是完全适用的。

看图 2.1，这是一个由网络组成的世界，在这个图中，整个世界都是由一个网络覆盖着的，人们通过网络进行互联，通过网络从事着各种事情，衣、食、住、用、行无不与网络有着密切的联系。

走进网络世界，你的一个简单的念头就会迅速地传到大洋彼岸的另一个人心里，你的一篇文章可以在转眼之间让全世界的人都知晓；走进网络世界，可看见远在千里发生的一切，可以听见世界上任何角落里的声音。在网络世界里，你的思维、观点、信息、资源及行为时刻都在被别人影响着，也在时刻影响着别人。

我们作为一个个独立个体的人，在现实世界中，每个人生活的环境、知识、阅历都是有限的，而网络的连线方式则让一个个孤岛般的电脑之间开始有了交通，而掌控电脑的人则可以通过网络的入口进入一个虚拟的网络世界。这个世界里的资源和信息是无限的，交流的范围和广度也是无限的，这种无限的虚拟世界就是网络，没有什么东西比网络更能具体地呈现出人与人之间相互交错的复杂关系。

2.1.2　繁星般的网络

网络世界的真实面目只能赋以抽象性的描述，而无法像快照一般将其真实地展现在桌面上。因为就整个互联网而言，网络的结构实在太复杂、太庞大。如果把满天的繁星当作全世界的计算机，把星星之间的光辉当作计算机之间互联的网线，那么现实世界的网络就如同宇宙间遍布的繁星一样，而错综复杂的网络连接就是它们彼此照耀的光辉，相互之间互连如同一张无限的天网。如图 2.2 模型化地展示了网络世界的面目。



图 2.1　由网络组成的世界

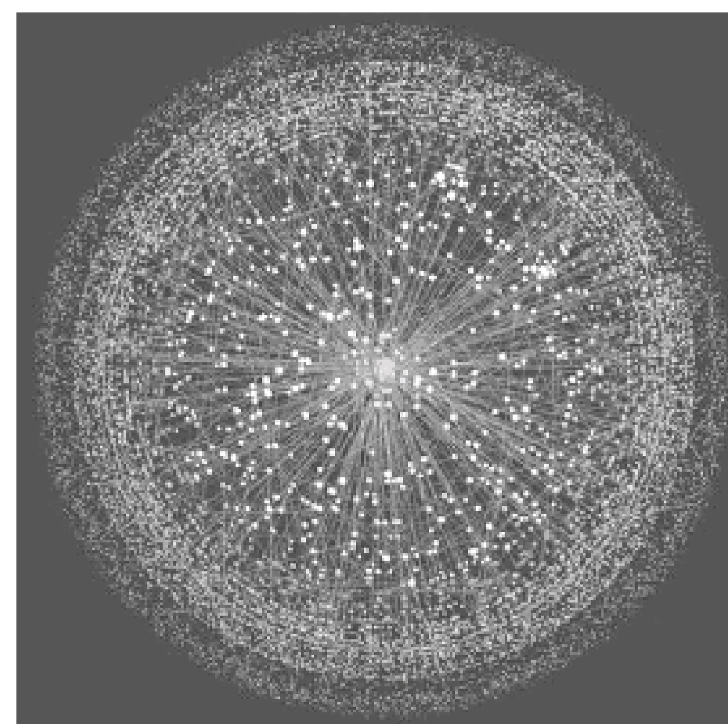


图 2.2　网络世界模型

图 1.2 展示的是网络世界的模型，而现实中的网络也正如同图 1.2 展示的一样，图中的每一个点就是现实中的一台计算机，彼此间通过网络连线错综复杂地交织在一起，形成了一个复杂的、有组织、有层次的、可连通的互连网状系统。

2.1.3　如何组织网络

上面已经说过了，在网络世界里，每个点就是一台计算机、就是一个人，每个点都接

入到这个无限的网络中，然后形成了结点之间无限的交织关系。理论上讲，在任何一个互联的网络中，从任何一个点出发，就可以遍历网络中所有的点，也就是说，网络中的任意一个点和其他的任何点之间都是连通的。

那么，要如何组织这些点才能达到这一目的呢？如果想要与 100 个人通信，是不是要从这一点出发建 100 个网线然后进行连接？如果是 1000 个呢？显然这是行不通过的。

在网络世界里有一种专门的网络拓扑技术来组织和管理网络的结构，以保证在这个结构中所有的点都能有效地连通、协同以完成某一网络功能。

在图 2.3 中，是一个简单的网络连通模型。在这个基于 WWW 的网络中，有 a、b、c、d、e 这 5 个点，通过不同的方式接入到网络系统中，它们之间并没有两两互联，但它们任何两点之间都是连通的。

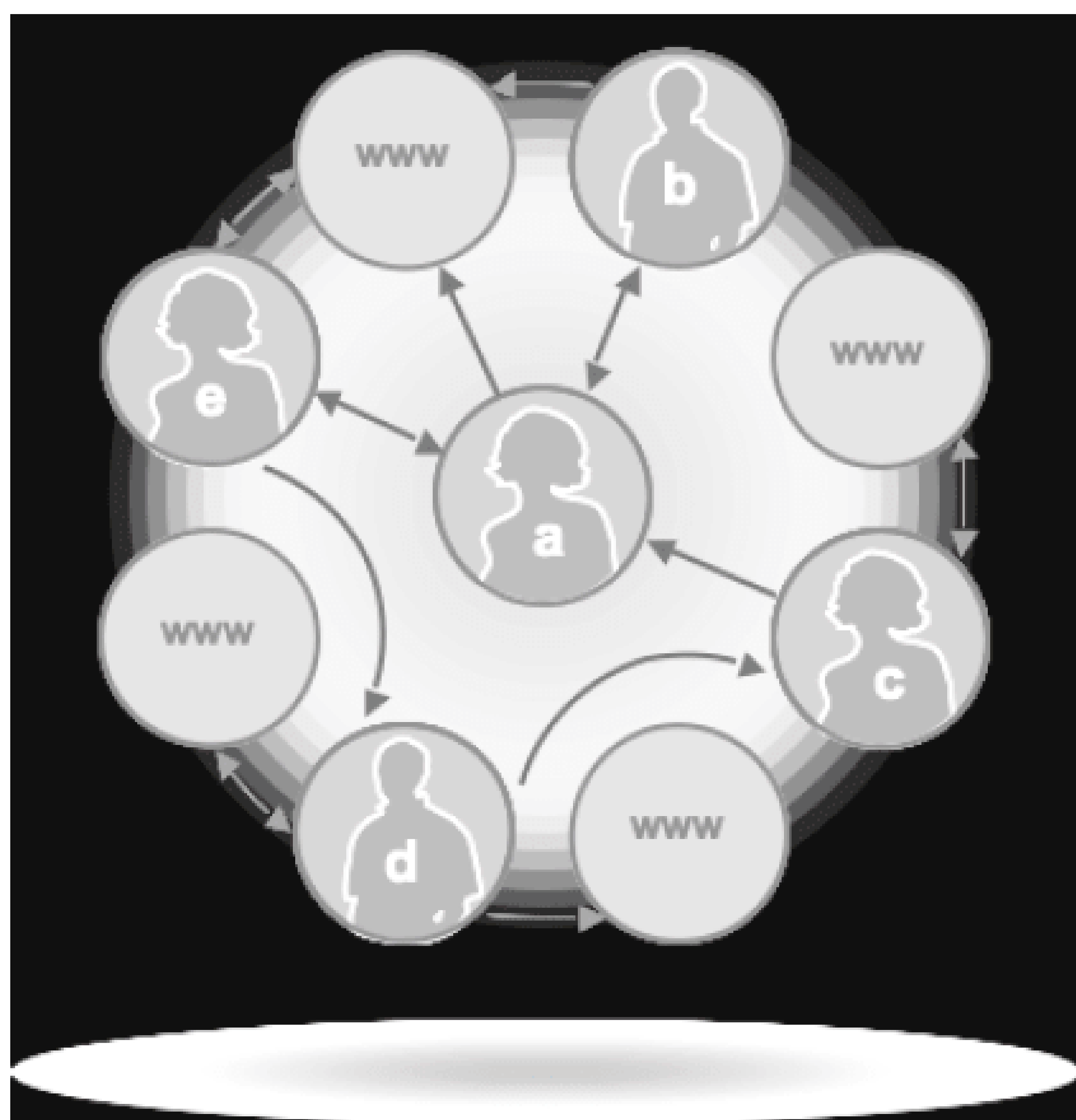


图 2.3　网络连通模型图

图 2.3 所展示的网络结构，是互联网系统中最基本的结构模型。网络中的点通过某种方式组织起来，以契约的（协议）形式作为建立和维持互联关系的基础，依靠外部机构（Web 服务器等）执行集中式的协调、组织、分发、转向等工作。被联结在这一结构中的各实体（客户端计算机）是相互独立的，彼此之间并没有所有关系或隶属关系，只是通过相对松散的契约纽带，通过一种互惠互利、相互协作、相互信任和支持的机制来进行密切的合作。通过这种方式，在单一的小网络的基础上，进行递归式的层层扩展和逐级扩散就会形成结构和组织都相对庞大及复杂的大网络。

以上所说的是对网络的组织和结构方式相对简单和抽象的描述，而在实际的网络组织过程中，还会涉及网络的物理连接、逻辑或虚拟设备的排列、结点间的配置等各种技术，这些技术统称为网络的拓扑技术，2.2 节将会详细讲述网络的拓扑相关的知识。

2.2　网络拓扑技术

在 2.1 节中，着重讲述了网络的整体结构模型和基本的互联关系，也引出了网络拓扑的概念，学习网络拓扑技术对理解网络的整体结构、对分析网络的效能和特点以及深入理

解网络的通信方式、传输方式都有重要的意义。本节就重点学习网络的拓扑技术。

2.2.1　什么是网络拓扑结构

简单地说，计算机的连接及组织所形成的结构就叫做“网络拓扑结构”（Topology）。在计算机科学中，网络拓扑结构指的是构成网络的成员间特定的物理的、或者逻辑的排列方式，直观上看，网络拓扑就是在计算机网络中传输媒体的互连的各种设备的物理布局，特别是计算机分布的位置及电缆如何通过它们。

注意： 网络拓扑仅由结点之间的连接配置所决定。结点之间的距离、物理互连、传输率或信号类型不作用在一个网络拓扑中。也就是说，如果两个网络的连接结构相同我们就说它们的网络拓扑相同，尽管它们各自内部的物理接线、结点间距离可能会有不同。

网络的拓扑结构主要是处理好网络设备及计算机之间的排列和连接，如果将这种排序用几何连接形状来表示，画成图就叫网络“拓扑图”，根据网络的不同结构和不同应用，网络的拓扑结构可以分很多种类型，如图 2.4 为最常用的网络拓扑结构图。

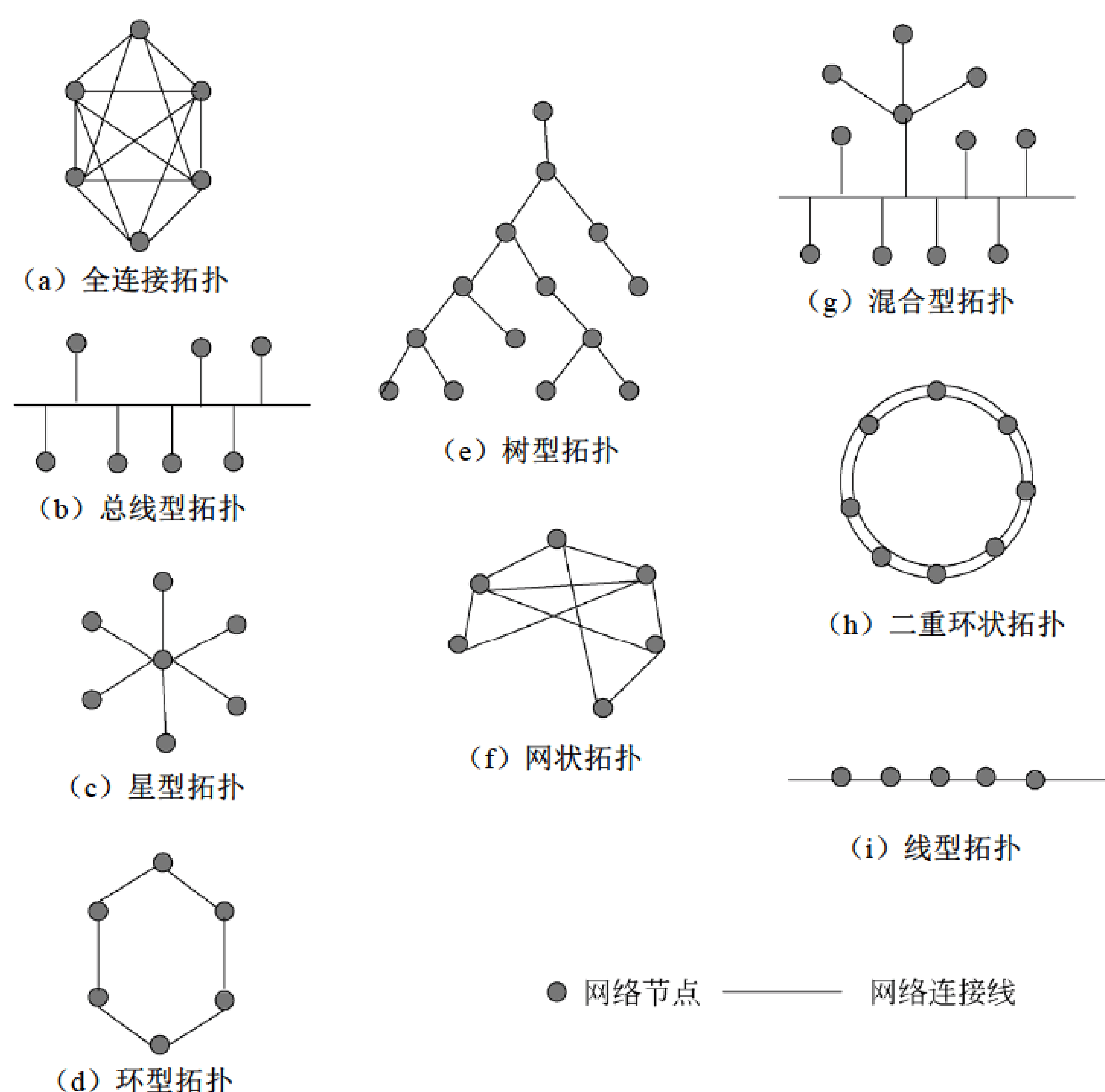


图 2.4　常见的网络拓扑结构图

图 2.4 中所展示的网络拓扑结构图中，从 a～i 依次是：（a）全连接网络拓扑，（b）总线型网络拓扑，（c）星型网络拓扑，（d）环型网络拓扑，（e）树型网络拓扑，（f）

网状网络拓扑，(g)混合型网络拓扑(此混合型网络拓扑是星型拓扑和总线型拓扑的混合)，(h) 二重环状网络拓扑， (i) 线型网络拓扑。

注意：网络拓扑图是研究和设计网络的基础，在网络规划、网络布线及网络分析中都有着十分重要的作用。

以上这些网络拓扑并不是孤立存在，在设计和构建一个网络的时候，通常是若干个拓扑结构混合应用在一起，形成更复杂的拓扑结构，在实际应用中，应根据网络的功能、应用领域、应用范围等因素选择正确的拓扑方式。

在当前的 Internet 中，应用最多的网络拓扑结构是星型网络拓扑结构、总线型和环型拓扑结构。图 2.5 是这 3 种拓扑结构的实物连接示意图，为后文讲述 P2P 网络拓扑的需要，下面就着重讲述一下这 3 种网络拓扑结构。

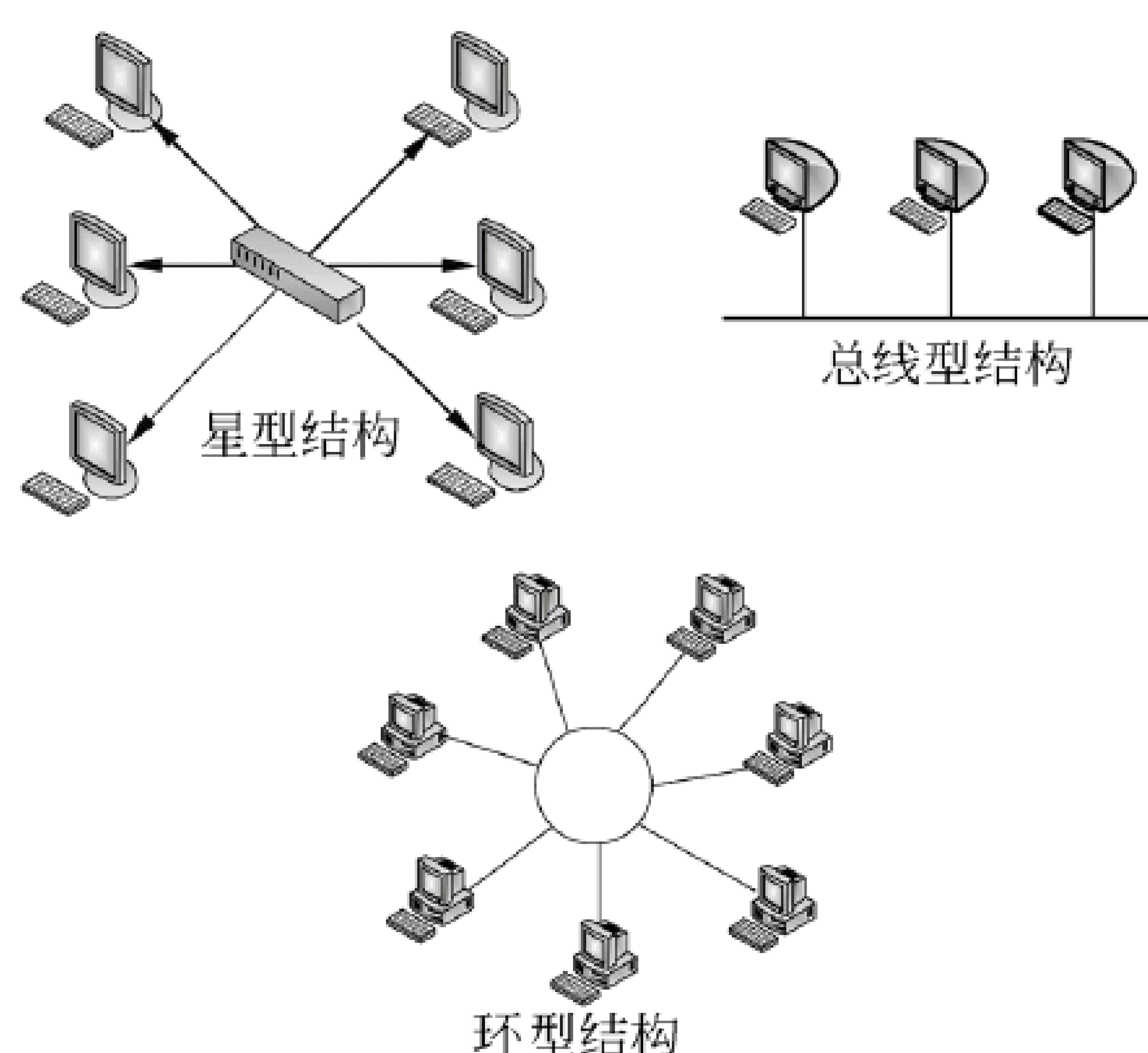


图 2.5　星型拓扑、总线型拓扑和环型拓扑实物连接图

2.2.2　星型拓扑结构

星型网络由中心结点和其他与中心结点连接的子结点（也叫从结点）组成，中心结点可直接与子结点通信，而子结点间必须通过中心结点才能相互通信。在星型网络中，中心结点通常由一种称为集线器或交换机的设备充当，因此网络上的计算机之间是通过集线器或交换机来相互通信的，星型网络拓扑是目前局域网中最常见的网络拓扑连接方式。图 2.6 为星型拓扑的网络连接示意图。

根据图 2.6 的结构而布建的网络就是拥有星型拓扑结构的网络，在构建星型的物理连接网络时，常使用双绞线连接，结构上以集线器（HUB）为中心，呈放射状态连接各台电脑。如图 2.7 所示为星型拓扑连接的实物图。

如图 2.7 就是一个星型的网络实物连接图，网络主体由中心结点和子结点组成，中心结点是一台交换机，其子结点都分别与中心结点连接，星型拓扑结构的网络有以下几个特点。

- 网络结构简单，便于管理（集中式）。
- 每台接入网络的主机均需物理线路与处理机互连，线路利用率低。

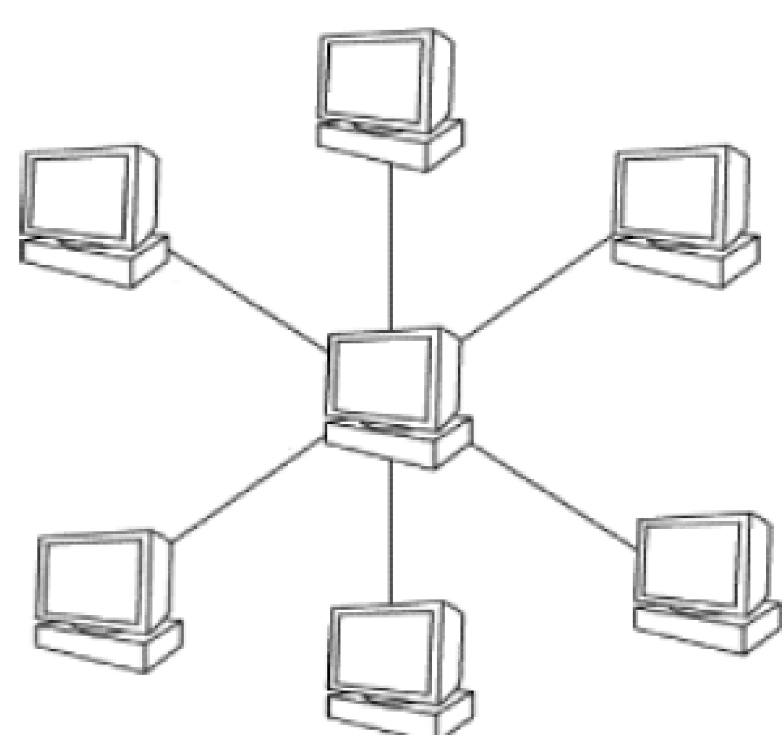


图 2.6　星型拓扑网络连接示意图

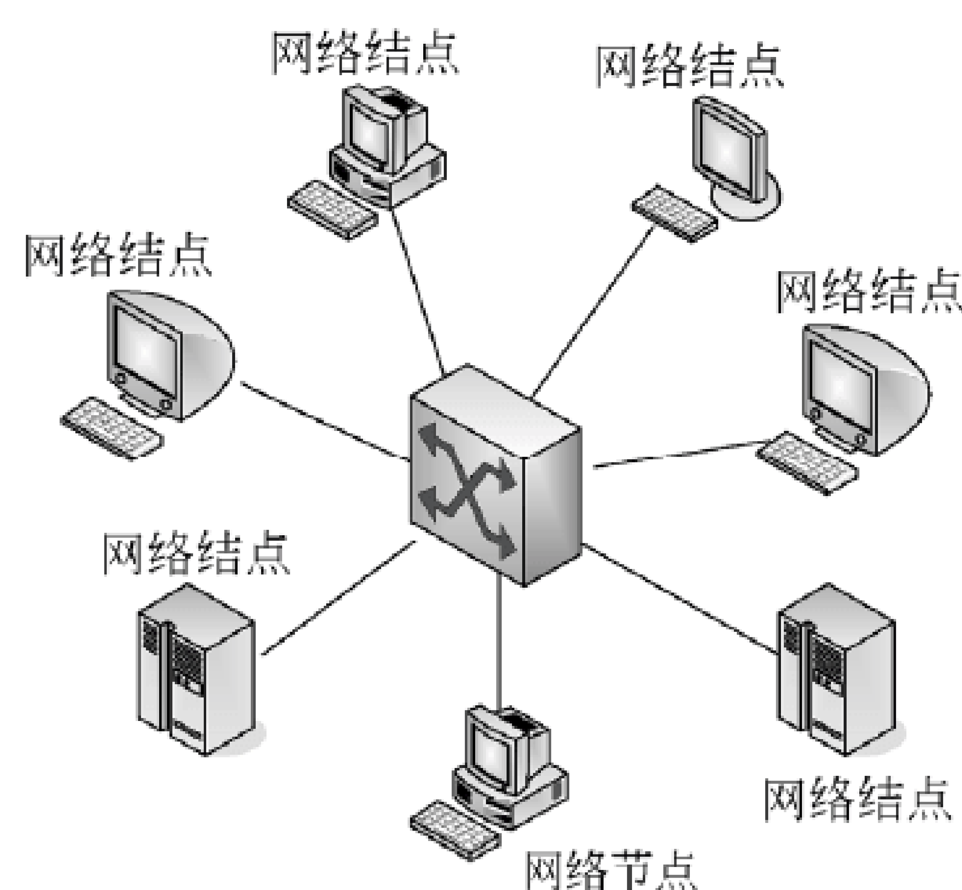


图 2.7　星型网络拓扑的实物连接图

- 处理机负载过重，它需要处理所有的服务，因为任何两台接入网络的主机之间交换信息，都必须通过中心处理机。
- 入网主机故障不影响整个网络的正常工作，中心处理机的故障将导致网络的瘫痪。

在星型拓扑结构的网络中，连接设备（如 HUB、交换机等）上会有许多指示灯，遇到故障时很容易发现出故障的电脑，而且一台电脑或线路出现问题不影响其他电脑，这样网络系统的可靠性大大增强。另外，如果要增加一台电脑，只需连接到 HUB 上就可以，很方便扩充网络，所以星形结构的网络现在非常流行。

注意： 星型网络结构根据其自身的特点，在实际应用中，局域网、广域网中最常见。

2.2.3　总线型拓扑结构

总线型网络拓扑是一种比较简单的计算机网络结构，它采用一条称为公共总线的传输介质，将各计算机直接与总线连接，信息沿总线介质逐个结点广播传送。如图 2.8 是总线型拓扑的网络连接示意图。

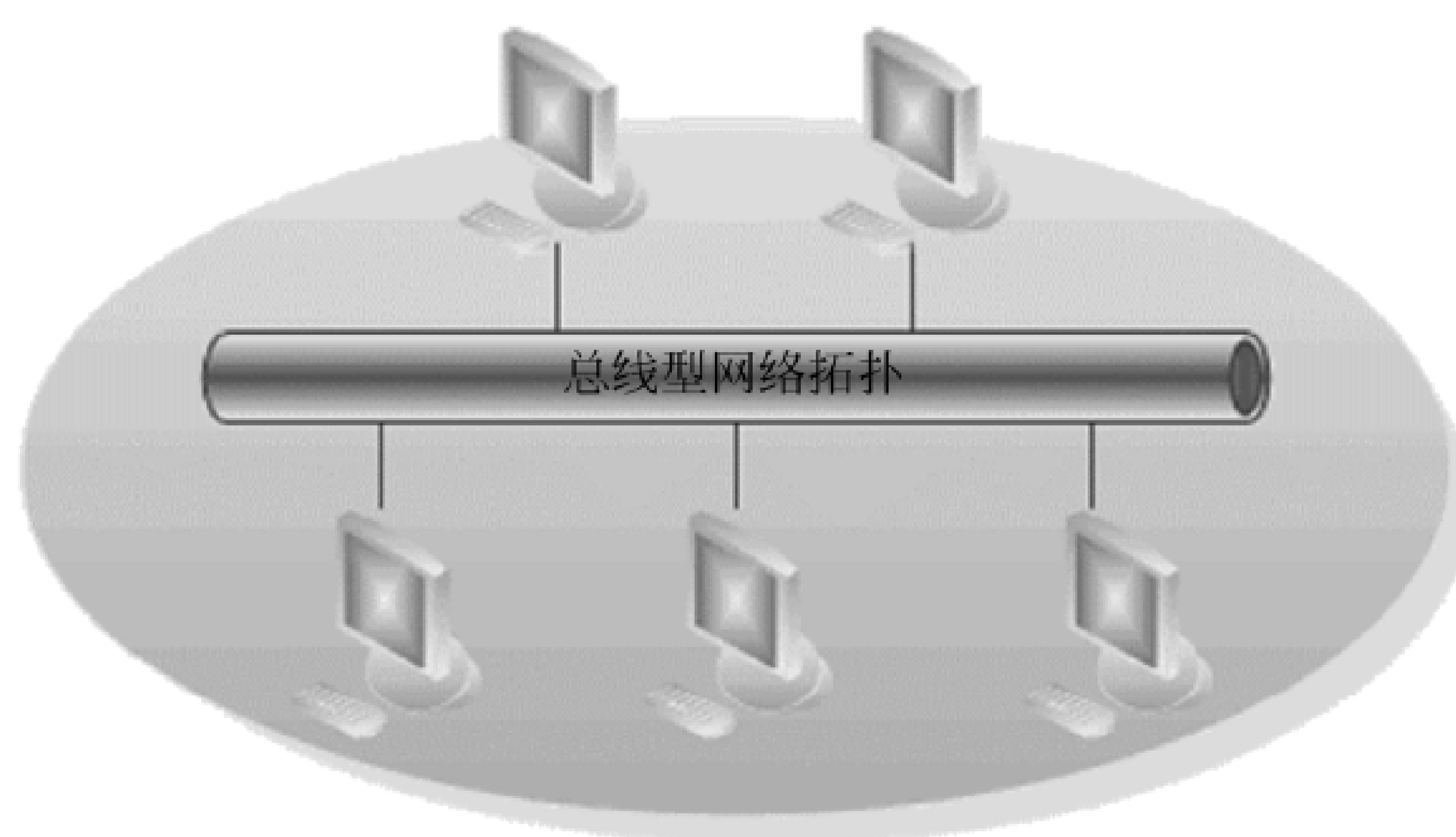


图 2.8　总线型网络拓扑结构示意图

在总线型网络拓扑结构中，所有入网设备共用一条物理传输线路，所有的数据发往同一条线路，并能够由连接在线路上的所有设备感知。入网设备通过专用的分接头接入线路。总线网拓扑是局域网的一种组成形式。

依据总线型拓扑构建的网络就叫做总线型网络，在具体的实物连接的过程中，一般需要用到 BNC 接口网卡、BNC－T 型接头、终结器和同轴细缆等网络器材。然后将所有电脑、打印机及其他用于互联的网络设备连接在一条线上，使用同轴电缆连接，就像一条线上拴着的很多只蚂蚱，总线型的实物连接图如图 2.9 所示。

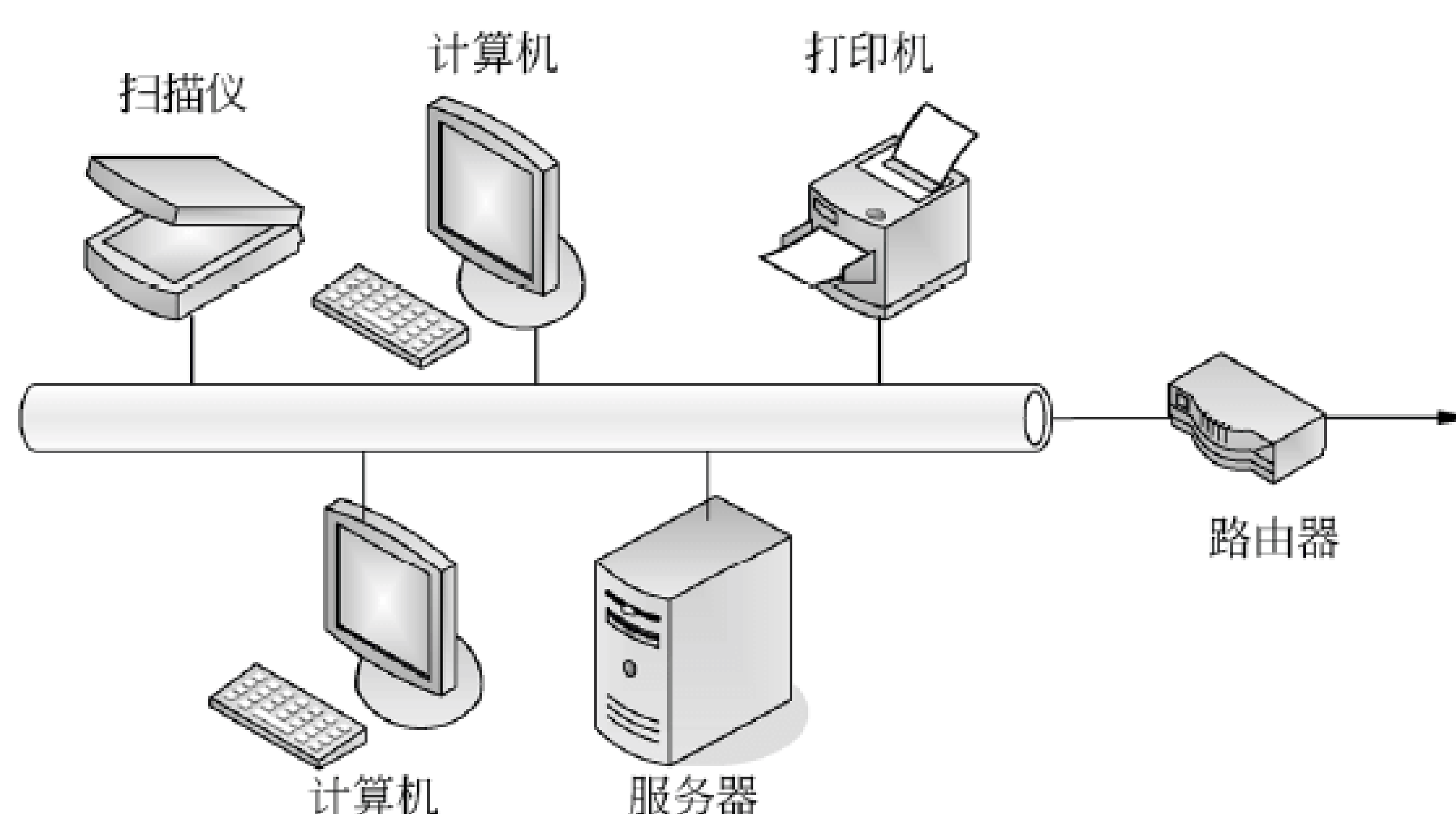


图 2.9　总线型网络拓扑的实物连接图

总线型网络在实际应用中一般有以下几个特点：

- 多台机器共用一条传输信道，信道利用率较高。
- 同一时刻只能有两台计算机进行通信。
- 某个结点的故障不影响网络的工作。
- 网络的延伸距离有限，结点数有限。

注意：总线型网络拓扑适合使用在电脑不多的局域网上或对实时性要求不高的环境中，因为电缆中的某段出了问题，其他电脑也无法接通，会导致整个网络瘫痪。

2.2.4　环型网络拓扑结构

环型网络拓扑是将计算机连成一个环状的结构，在这个环状的网络结构中，入网设备通过转发器接入网络，每个转发器仅与两个相邻的转发器有直接的物理线路。环形网的数据传输具有单向性，一个转发器发出的数据只能被另一个转发器接收并转发。所有的转发器及其物理线路构成了一个环状的网络系统。如图 2.10 所示，是一个环状的网络拓扑示意图。

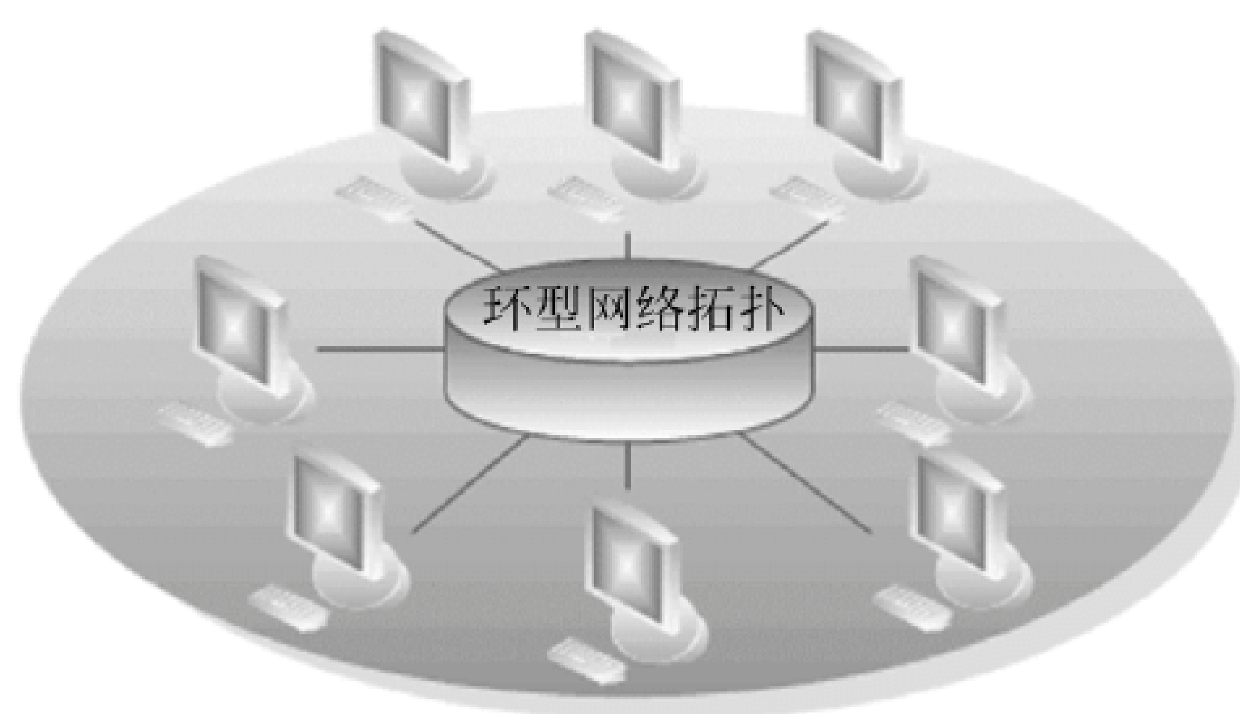


图 2.10　环型网络拓扑结构示意图

在一个环型的网络拓扑结构中，数据信号以“接力”的方式传输。如图 2.11 所示，箭头方向为数据的传输方向。

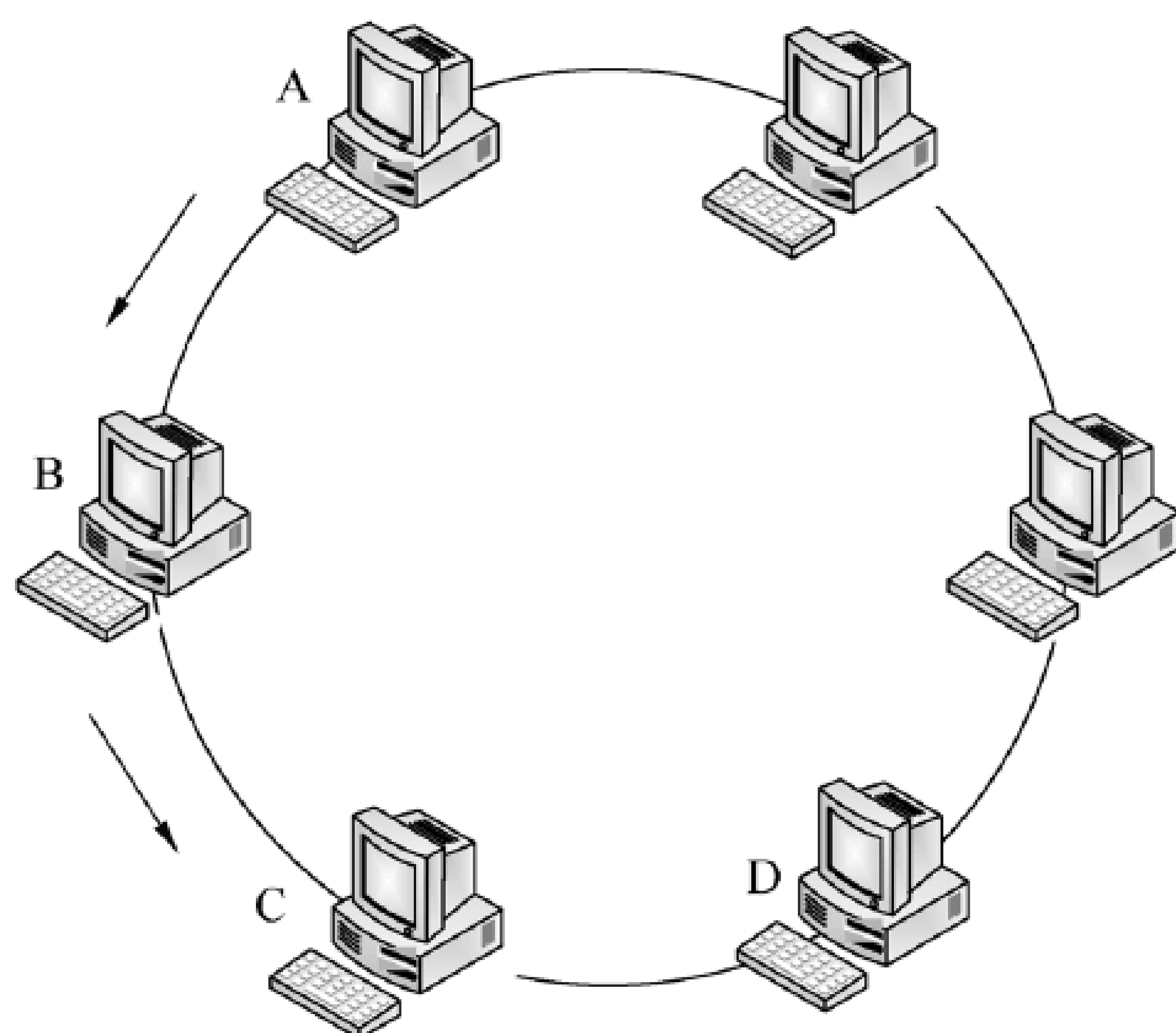


图 2.11　环型网络拓扑的实物结构及信息传递示意图

若计算机 A 欲将信号传输给计算机 D 时，首先它会将信号传送给计算机 B，计算机 B 收到信号后发现不是给自己的，于是将这个信号再传给计算机 C。C 接到信号后会执行与 B 同样的动作，检查到该数据也不是自己所需要，于是再将它传给 D，这样，数据便通过这种方式传送到计算机 D 了。环型网络拓扑中的信号都是以这种方式传递的。

环形网络拓扑一般应用在局域网，或实时性要求较高的网络环境中，环型网络具有以下特点。

- 实时性较好（信息在网中传输的最大时间固定）。
- 每个结点只与相邻两个结点有物理链路。
- 传输控制机制比较简单。
- 某个结点的故障将导致整个网络瘫痪。
- 单个环网的结点数有限。

注意： 在实际应用中，上述 3 种类型的网络经常被综合应用，并形成因特网。因特网是指将两个或两个以上的计算机网络连接而成的更大的计算机网络。

2.3　基于拓扑结构的 P2P 网络分类

在 2.2 节中已经讲过了网络拓扑结构的相关知识，对 P2P 网络而言，P2P 网络有其独特的拓扑结构，在 P2P 网络中，结点之间的拓扑结构是确定系统类型的重要依据。同时，P2P 面临的最大挑战之一也是如何在没有中心服务器的模式下维护网络拓扑结构，以实现 P2P 路由和内容搜索。因而理解并掌握 P2P 的网络拓扑结构及其分类是学习 P2P 的基础。

在研究 P2P 网络的时候，一般要构造一个非集中式的拓扑结构，在构造的过程中需要解决网络中大量结点如何命名、组织以及确定结点的加入或离开方式、出错恢复等问题。

根据拓扑结构的关系可以将 P2P 网络拓扑结构分为 4 种形式：

- 集中式拓扑（Centralized Topology，也称作中心化拓扑）。
- 全分布式结构化拓扑（Decentralized Structured Topology，也称作 DHT 网络）。
- 全分布式非结构化拓扑（Decentralized Unstructured Topology）。
- 混合式拓扑（Partially Decentralized Topology，也称作中心化拓扑）。

以上这 4 种 P2P 的网络结构，分别于 P2P 发展的不同时期出现，因而在其分类过程中，也可以按 P2P 发展的“代”来划分。集中式 P2P 网络常对应于第一代 P2P，而第二代 P2P 则是全分散分布式网络体系结构，也叫纯 P2P 网络结构，第三代 P2P，就是混合式的 P2P 网络结构，在这种划分中，把当前正在发展的 P2P 技术归于第四代。P2P 的分类和分代，可以用图 2.12 来表示。

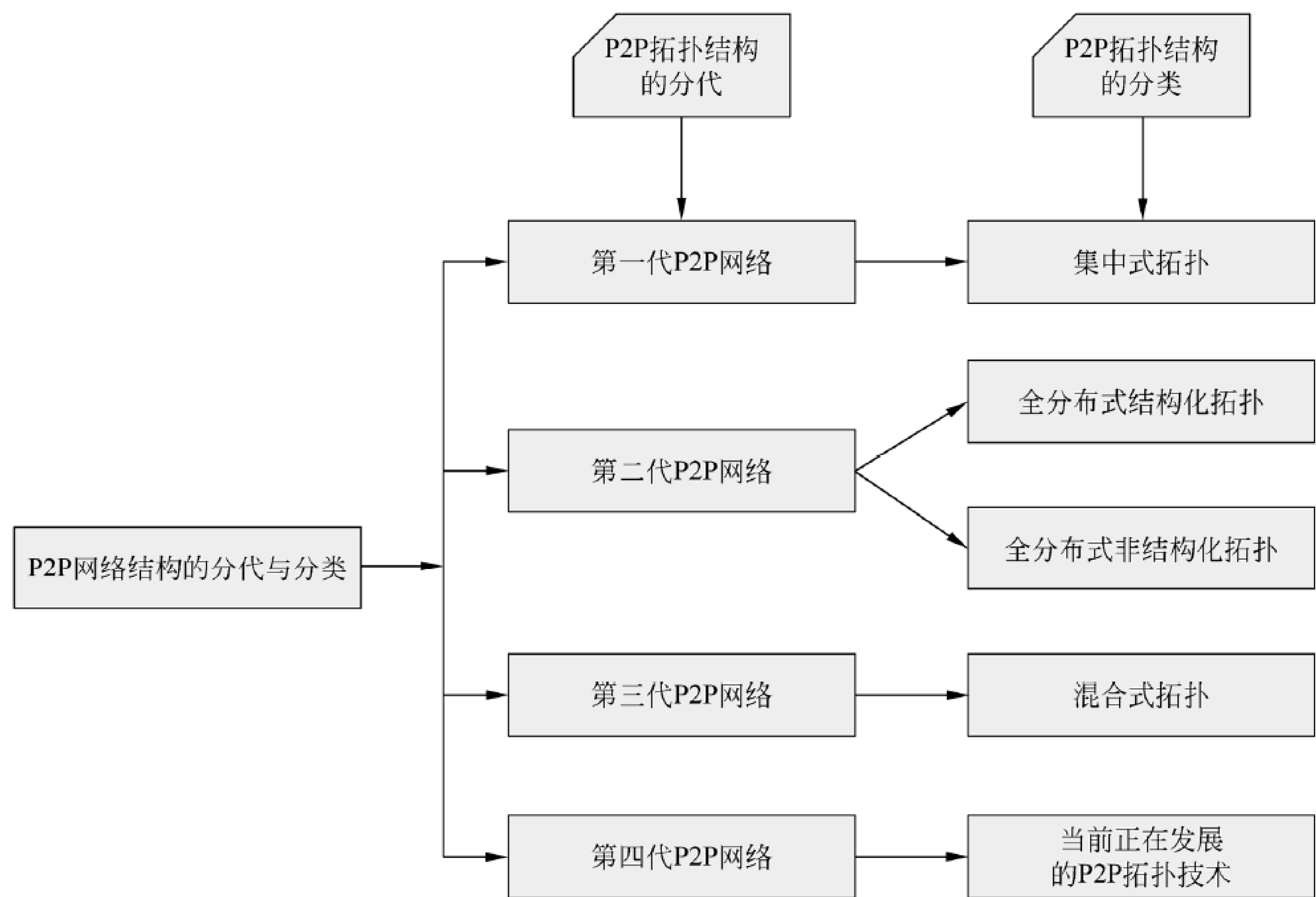


图 2.12　P2P 网络拓扑结构的分代与分类图

2.4　集中式的 P2P 网络拓扑

集中式的 P2P 网络拓扑也称为中心化的 P2P 网络结构，在 P2P 发展的过程中是最早出现的 P2P 网络拓扑结构模型，因而也称为第一代 P2P 网络。集中式的 P2P 网络拓扑是基于中央控制的网络体系结构，在研究 P2P 网络结构、P2P 搜索技术方面有着重要的意义，下面就详细讲解集中式 P2P 网络结构的相关知识。

2.4.1　集中式 P2P 网络拓扑的原理

集中式 P2P 网络，类似于一个抽象的“星形网络拓扑”，由一个中央服务以星状的形式与各客户机连接，但这种连接并不是物理上的星状拓扑，而是基于集中式 P2P 网络拓扑协议而形成的一个虚拟的星状结构。

在集中式 P2P 网络中，网络的主体由一个处于中心地位的索引目录服务器和连接到目录服务器的各结点客户机组成，目录服务器用来管理和组织 P2P 的各客户端结点。

在网络运行过程中，P2P 结点向中央目录服务器注册关于自身的信息（名称、地址、资源和元数据等），但这些注册的所有内容都分散式地存储在各个结点中而非并服务器上，查询结点根据目录服务器中信息的查询以及网络流量和延迟等信息，来选择与定位其他对等点并直接建立连接，而不必经过中央目录服务器进行。

2.4.2　集中式 P2P 网络结构

集中式的 P2P 网络结构如图 2.13 所示。组成这个网络需要一个目录服务器和若干 Peer 结点，图 2.13 中，目录服务器为 Directory Server，相对应的 Peer 结点分别为 PeerA、PeerB、PeerC、PeerD 等。

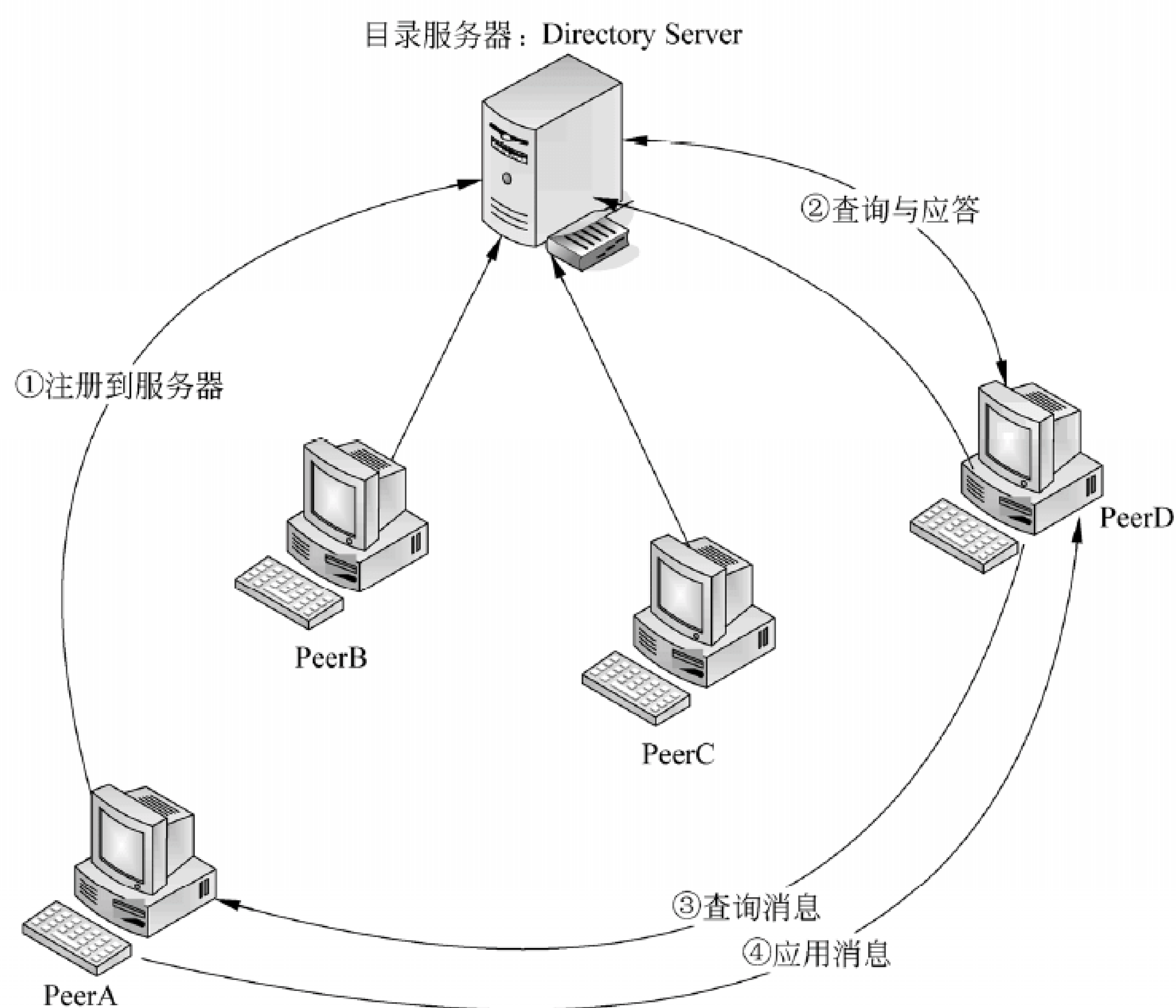


图 2.13　集中式 P2P 网络模型结构图

在集中式的 P2P 网络结构中，由目录服务器管理和组织结点信息，各个对等的客户端，也就是 Peer 结点要加入到 P2P 系统中的时候，都要登录到中心目录服务器（也叫索引服务

器或中心服务器）上，如图 2.13 中的 Directory Server。

在这种集中式的 P2P 网络中，中央服务器用来保存所有 Peer 结点用户上传的文件索引和存放位置的信息，以及拥有此文件结点自身的信息，当某个用户需要某个文件时，首先连接到中央服务器，再对服务器进行检索，并由服务器返回存有该文件的用户信息；再由请求者直接连到文件所有者所在的结点主机上，与此主机建立连接以传输文件。

从 Peer 结点注册目录服务器到向目录查询资源的整个过程，如图 2.13 中所示，需要如下 4 个步骤。

- ❑ 当有 PeerA 加入到集中式的 P2P 网络时，需要向目录服务器进行注册（Registration），将自身的各种相关信息注册到目录服务器中。
- ❑ 网络中的 PeerD 要搜索某一 PeerA 的文件 File 时，PeerD 向目录服务器发送查询请求，目录服务器将 PeerA 注册的信息，包括文件 File 的信息以应答的方式返回给 PeerD。
- ❑ PeerD 根据 PeerA 的信息（IP 地址、端口等）与 PeerA 建立连接，并向 PeerA 发送对文件 File 的请求。
- ❑ PeerA 将自身的文件 File 返回 PeerD，此时，PeerA 与 PeerD 就建立了联通关系，相互之间进行通信，而目录服务器则不再参与 PeerA 与 PeerD 的通信过程。

以上是在集中式 P2P 中 Peer 结点的加入和结点进行文件查询时的过程，这一过程在实际应用中既有优点也有不足。

2.4.3　集中式 P2P 网络的优缺点

前面介绍了集中式 P2P 网络的工作原理及其基本的通信过程，这种集中式 P2P 网络最大的优点是维护简单，有效提高了网络的可管理性。同时，在资源的搜索与发现方面，由于资源的发现依赖中心化的目录系统，发现算法灵活、高效并能够实现复杂查询，使得对共享资源的查找和更新非常方便，资源发现效率高，这是集中式 P2P 的优点。

集中式 P2P 也存在一些问题，最大的问题与传统客户机/服务器结构类似，容易造成单点故障。总结起来，这种 P2P 网络结构易出现的问题主要表现在以下几个方面。

- ❑ 中央服务器的瘫痪容易导致整个网络的崩溃，可靠性和安全性较低。如果该服务器失效，整个系统都会瘫痪。
- ❑ 随着网络规模的扩大，当用户数量大量增加时，系统的性能会大大下降。对中央索引服务器进行维护和更新的费用将急剧增加，所需成本过高。
- ❑ 中央服务器的存在，容易引起共享资源在版权问题上的纠纷。

总之，从 P2P 的实质意义上讲，集中式 P2P 网络非纯粹意义上的 P2P 网络模型。对小型网络而言，集中目录式模型在管理和控制方面占一定优势。但鉴于其存在的种种缺陷，该模型并不适合大型的 P2P 网络应用。

在 P2P 的发展历程中，第一代 P2P 网络普遍采用的就是这种集中式结构模式，无论集中式的 P2P 存在多少问题，它曾经引领了一代 P2P 技术的发展，至今也仍有应用，最经典的案例就是著名的 MP3 共享软件 Napster。

2.4.4　集中式 P2P 网络系统实例

Napster 是最早出现的 P2P 系统之一，并在短期内迅速成长起来。它实质上并非是纯粹的 P2P 系统，而是通过一个中央索引服务器保存所有 Napster 用户上传的音乐文件索引，以及存放位置的信息，这种结构模型正是集中式 P2P 网络所具备的特点。它的工作原理也与集中式 P2P 网络一样。

注意：关于 Napster 的相关知识，在第 1 章中已有讲述，读者可自行参考第 1 章的有关内容。

在 Napster 模型中，一群高性能的中央服务器保存着网络中所有活跃的、对等计算机上共享资源的目录信息。当需要查询某个文件时，对等机会向一台中央服务器发出文件查询请求。中央服务器进行相应的检索和查询后，会返回符合查询要求的对等机地址信息列表。查询发起的对等机接收到应答后，会根据网络流量和延迟等信息进行选择，找到最佳的、合适的对等机并与之建立连接，如是两个对等机之间开始进行文件传输。Napster 拓扑结构如图 2.14 所示。

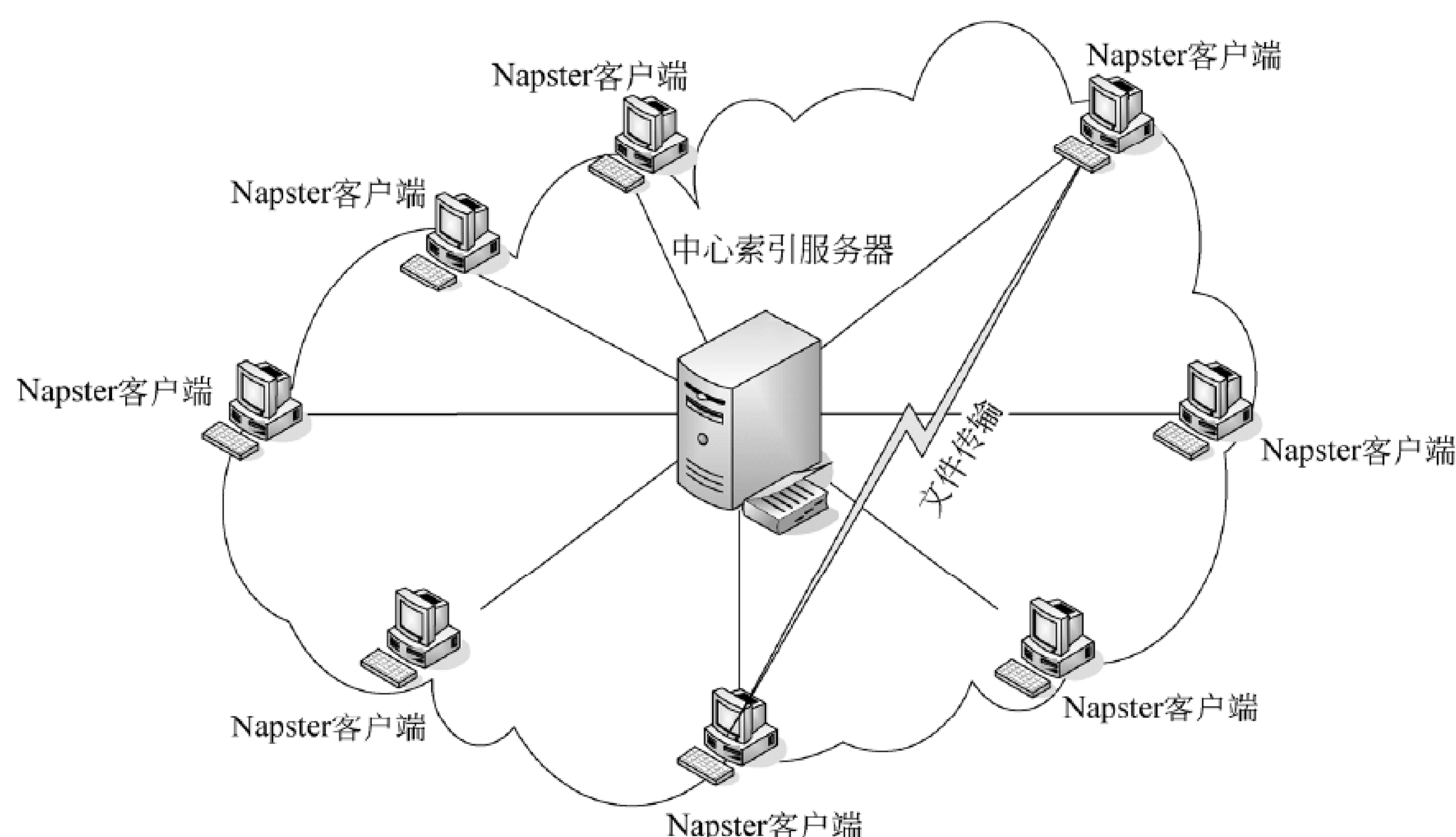


图 2.14　Napster 的拓扑结构图

Napster 实现了文件查询与文件传输的分离，有效地节省了中央服务器的带宽消耗，减少了系统的文件传输延时。

Napster 是基于集中式 P2P 网络拓扑构建的，自然就不可避免地存在着集中式 P2P 拓扑的隐患，在 Napster 中，如果中央的目录服务器失效，整个系统都会瘫痪。当用户数量增加到 10^5 或者更高时，Napster 的系统性能会大大下降。另一个问题在于安全性上，Napster 并没有提供有效的安全机制。

注意： Napster 还存在版权问题，在 Napster 的中央目录服务器上存着大量 MP3 文件，这些文件大部分没有经过官方的授权，因而很容易引起版权纠纷。

2.5　全分布式结构化 P2P 网络拓扑

全分布式 P2P 网络结构也被称作广播式的 P2P 网络结构模型，它包括两种类型，一种是全分布式结构化的 P2P 网络，另一种则是全分布式非结构化的 P2P 网络，本节将重点讲解其中的一类，就是全分布式结构化的 P2P 网络。

2.5.1　什么是 DHT 技术

全分布式结构化拓扑的 P2P 网络主要是采用分布式散列表技术来组织网络中的结点。分布式散列表的全称是 Distributed Hash Table，简写成 DHT，要学习分布式的结构化 P2P 网络拓扑，就必须要了解 DHT 技术。

DHT，是分布式计算系统中的一类，用来将一个关键值（key）的集合分散到所有在分布式系统中的结点上，并且可以有效地将信息转送到唯一一个拥有查询者提供的关键值的结点（Peers）。这里的结点类似散列表中的储存位置。分布式散列表通常是为了拥有极大结点数量的系统，而且系统的结点常常会加入或离开（例如网络断线）而设计。分布式散列表示意图，如图 2.15 所示。

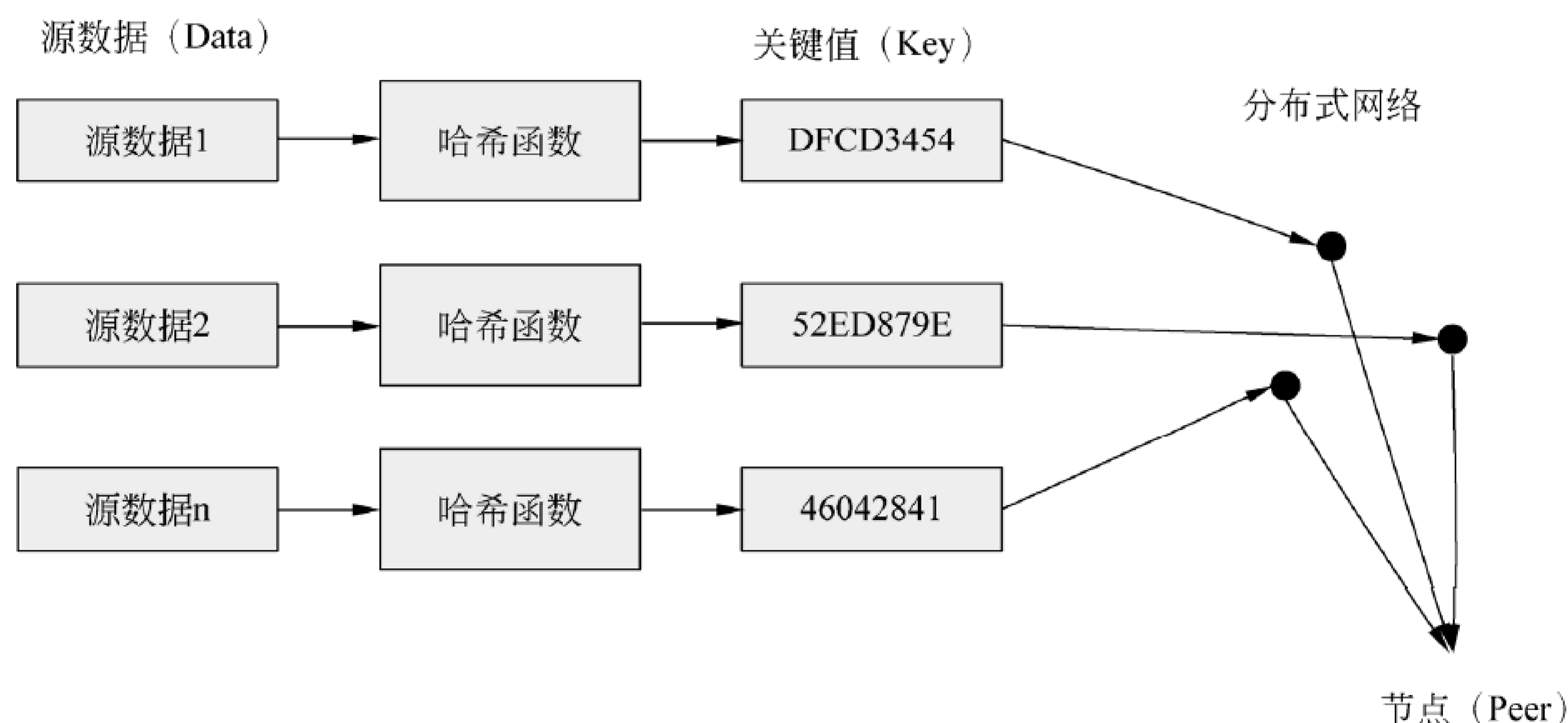


图 2.15　分布式散列表结构示意图

在一个结构性的延展网络（overlay network）中，参加的结点需要与系统中一小部分的结点沟通，这也需要使用分布式散列表。分布式散列表可以用以建立更复杂的服务，例如分布式文件系统、点对点技术的文件分享系统、多播、任播（any cast）、网域名称系统以及即时通信等。

2.5.2　DHT 的相关概念

要理解 DHT 的原理，首先需要明白两个基本的概念，一个是关键值空间分割，另一个是延展网络，下面分别讲一下这两个概念。

1．关键值空间分割

关键值分割是 DHT 中最基本的思想，大多数的分布式散列表使用某些稳定散列（consistent hashing）方法来将关键值对应到结点中。此方法使用了一个函数，将这个函数描述为 δ(k1，k2)，通过这个函数来定义一个抽象的概念，它所表示的意思是，从关键值 k1 到 k2 的距离。

在基于 DHT 的 P2P 网络中，每个结点都被指定了一个关键值，称为 ID。ID 为 i 的结点可以根据函数 δ 计算出最接近 i 的所有关键值。比如，δ(i，j)表示 ID 为 i 的结点到 ID 为 j 的结点之间的距离，这个距离是可量度的抽象概念。

例如，在 Chord 网络中（关于 Chord 网络，在下文会讲到，这里只需简要了解），分布式散列表将网络结点的关键值视为一个圆上的点。这样，通过一个圆上所有的点就描述了整个关键值空间，而函数 δ(k1，k2)则是沿着这个虚拟的圆顺时针地从 k1 走到 k2 的距离。通过这种方式，圆形的关键值空间就被切成连续的圆弧段，而每段的端点都是结点的 ID。如果 i1 与 i2 是邻近的 ID，则 ID 为 i2 的结点拥有落在 i1 及 i2 之间的所有关键值。

稳定散列拥有一个基本的性质：增加或移除结点只改变邻近 ID 的结点所拥有的关键值集合，而其他结点则不会被改变。对比于传统的散列表，若增加或移除一个位置，则整个关键值空间就必须重新对应。

2．延展网络

DHT 中另一个重要的概念就是延展网络，在分布式的 P2P 网络中，每个结点自身都存储着一些到其他结点（它的邻居）的连接信息，将这些包含着连接信息的结点总合起来就形成延展网络。而这些连接是使用一个结构性的方式来挑选的，这也是分布式结构化拓扑中所谓结构化的意义所在。

注意：以上分别描述的关键值空间分割及延展网络的基本概念，在大多数的分布式散列表实例中是相同的，但设计的细节部分则大多不同。

2.5.3　DHT 的原理与性质

分布式散列表的结构可以分成几个主要的元件。其基础是一个抽象的关键值空间（key space），例如说所有 160 位元长的字符串集合。关键值空间分割（key space partitioning）将关键值空间分割成多个并指定到在此系统的结点中。而延展网络则连接这些结点，并让它们能够借助在关键值空间内的任一值找到拥有该值的结点。

当这些元件都准备好后，可以用如下实例来描述用分布式散列表储存与读取的方式。

（1）假设关键值空间是一个 160 位元长的字符串集合。为了在分布式散列表中储存一

个文件，名称为 filename 且内容为 data。

（2）计算出 filename 的 SHA1 散列值，一个 160 位元的关键值 k，将此散列值信息执行一个 Put 操作，也就是 put（k，data），意思是：此信息送给分布式散列表中的任意参与结点。

（3）此信息在延展网络中被转送，直到抵达在关键值空间分割中被指定负责储存关键值 k 的结点，而（k，data）即储存在该结点上。

（4）其他结点若需要请求 data 时，只需要重新计算 filename 的散列值 k，然后执行一个 Get 操作，即 get(k)，送出信息给分布式散列表中的任意参与结点，以此来找与 k 相关的资料。

（5）此信息也会在延展网络中被转送到负责储存 k 的结点。而此结点则会负责传回储存的资料 data 给请求结点。

所有的分布式散列表都有某些基本的性质：对于任一关键值 k，某个结点或者就拥有 k，或者就拥有一个连接能连接到距离较接近 k 的结点。因此使用一般的贪心算法即可容易地将信息转送到拥有关键值 k 的结点。具体操作是：在每次执行时，将信息转送到 ID 较接近 k 的邻近结点，若没有这样的结点，那一定抵达了最接近 k 的结点，也就是拥有 k 的结点。这样的转送方法有时被称为“基于关键值的转送方法”。

除了基本的转送正确性之外，还有另外两个关键的限制：其一为保证任何的转送路径长度必须尽量短，因而请求能快速地被完成；其二为任一结点的邻近结点数目（又称最大结点度（Degree）必须尽量少，因此维护的花费不会过多。当然，转送长度越短，则最大结点度越大。以下列出常见的最大结点度及转送长度（n 为分布式散列表中的结点数）。

- 最大结点度 O(1)，转送长度 O(log n)；
- 最大结点度 O(log n)，转送长度 O(log n /log log n)；
- 最大结点度 O(log n)，转送长度 O(logn)；
- 最大结点度 O(n1/2)，转送长度 O(1)。

在以上 4 种转送方式中，第 3 个选择最为常见。虽然它在最大结点度与转送长度的取舍中并不是最佳的选择，但这样的拓扑允许较为有弹性地选择邻近结点。许多分布式散列表都利用这种弹性来选择延迟较低的邻近结点。

注意： 以上所讲内容有些是与算法设计相关的知识，读者如不明白可参考计算机数据结构与算法的相关知识点。

2.5.4　全分布式结构化 P2P 的原理

前面已经说过，全分布式结构化拓扑的 P2P 网络是采用分布式散列表技术来组织网络中的结点的，那么全分布式 P2P 网络的实现方式也正是 DHT 技术的完美应用。

通过 DHT 技术，将广域范围内大量的结点共同形成并维护一个巨大散列表。散列表被分割成不连续的块，每个结点被分配给一个属于自己的散列块，并成为这个散列块的管理者。利用 DHT 的原理，它可以提供 P2P 网络中分布于许多结点中数据的总体视点，独立于实际位置。因此，数据的位置依赖于当前的，DHT 状态而不固有地依赖于数据本身。

在 P2P 网络中，DHT 类结构能够自适应结点的动态加入/退出，有着良好的可扩展性、

鲁棒性、结点 ID 分配的均匀性和自组织能力。由于重叠网络采用了确定性拓扑结构，DHT 可以提供精确的发现。只要目的结点存在于网络中，DHT 总能发现它，发现的准确性也会得到保证。

全分布式结构化的 P2P 网络拓扑，有众多经典的实现案例，理解了这些案例也就理解了全分布式结构化的 P2P 网络拓扑的实质意义。最经典的案例有以下几种：

- 内容可寻址网络（Content addressable network，CAN）；
- Chord（Chord project）；
- Pastry（Pastry (DHT)）；
- Tapestry (DHT)（Tapestry (DHT)）。

以上这 4 种案例皆同时于 2001 年发表。从那时开始，相关的研究便一直十分活跃。在学术领域以外，分布式散列表技术已经被应用在 BT 及 CoralCDN（Coral Content Distribution Network），下面分别介绍一下这几种技术。

2.5.5　CAN 项目

CAN 的全称是 Content Addressable Networks，内容可寻址网络，是由 AT&T 所提出的点对点搜寻算法，CAN 利用多维坐标空间概念来建构的点对点架构。图 2.16 就是一个包含 5 个结点二维坐标系统的 CAN 架构概念图。

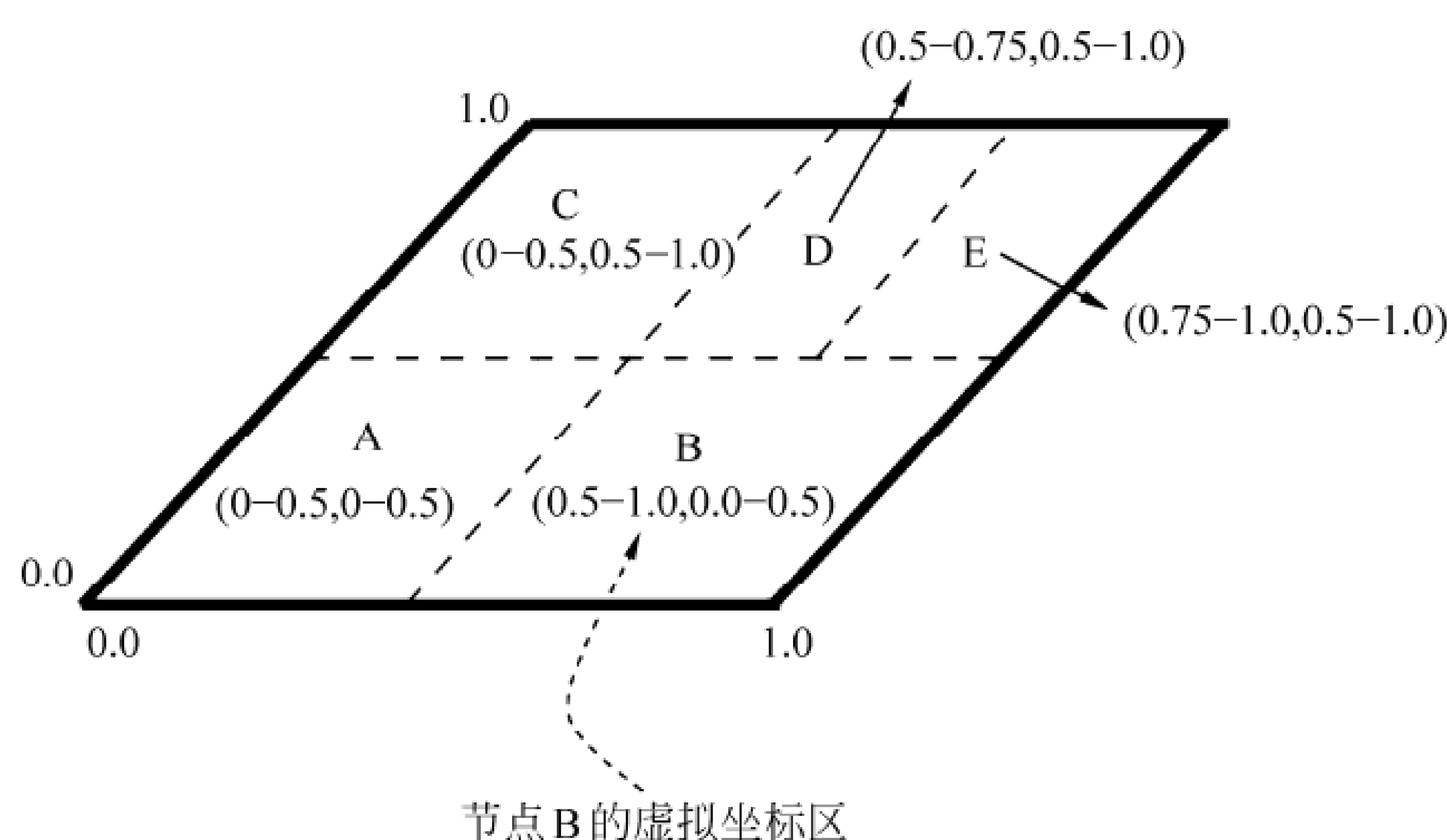


图 2.16　二维坐标系统的 CAN 架构图

图 2.16 中，结点 A 所拥有的坐标空间为（0–0.5，0–0.5），结点 B 所拥有的坐标空间为（0.5–1.0，0–0.5）。下面根据这个图，来说明一下在 CAN 系统中点对点建立的算法。

1．CAN 系统中建立点对点关系的算法

（1）CAN 将结点通过 DHT 技术映射到一个 n 维的笛卡儿空间中，并为每个结点尽可能均匀地分配一块区域。

（2）CAN 采用的散列函数通过对（key，value）键值对中的 key 进行散列运算，得到笛卡儿空间中的一个点，并将（key，value）键值对存储在拥有该点所在区域的结点内。

（3）当一个新的结点欲加入 CAN 系统时，新加入的结点会透过一个起始点

（Bootstrap）随机地选择系统中的结点。

（4）当这个随机的结点选择成功后，新加入的结点就将自己加入（JOIN）系统的相关信息传送给随机选择的结点。

（5）当被选择到的结点收到新结点发送来的加入信息时，则均分其所拥有的坐标空间。

以上的算法过程，结合图 2.16 所示，当有新的结点 E 加入到结点 D 的区域时，则 D 将其所拥有的坐标空间均分给 E。

注意：D 原有的坐标空间是（0.5—1.0，0.5—1.0），当 E 结点加入 D 区域后，D 就会将自己的坐标空间均分给 E，此时 D 的坐标空间就变成了（0.5—1.0，0.5—1.0），而 E 的坐标空间就是（0.5—1.0，0.5—1.0）。这里用到了笛卡儿空间的相关知识，读者可自行查阅相关资料。

以上就是在 CAN 系统中结点加入的方式。另一个重要的概念就是 CAN 系统中资源的存储方式，因为在 P2P 系统中，不但要管理结点的加入和退出，还要处理资源存储与搜索。下面就说明一下 CAN 系统中对资源的处理方式。

2．CAN 系统中对资源的处理方式

（1）当结点欲将新的资源加入到 CAN 系统中分享时，CAN 系统将其资源文件名依照哈希函数计算出一个坐标，并将此资源信息储存在此坐标空间的结点上。

（2）当有另一个结点欲搜寻这个资源时，CAN 利用每个结点所拥有的路由表（coordinate routing table）来搜寻资源信息。路由表内所储存的数据为记录在坐标空间中邻结点的 IP 地址及所拥有的空间信息。

3．CAN 系统中的搜索过程

（1）当起始点（上文提到的 Bootstrap）收到系统中结点要求搜寻资源的信息时，起始点会先利用哈希函数计算出此资源所代表的坐标，起始点会从系统中任意选择一个结点，并将搜寻资源的坐标送给被起始点所选择的结点。

（2）被选择到的结点收到资源数据信息时，会先查询资源数据是否存在于结点中，如果存在，则回报资源信息给提出搜寻资源的结点。如果不存在，则结点会依照结点中的路由表，依据贪心算法（greedy algorithm）找出一个与资源坐标最接近的结点，并转送此查询信息，依照此搜寻方式直到找到资源为止。

2.5.6　Chord 协议

Chord 是 2001 年由麻省理工学院提出的，其核心思想就是要解决在 P2P 应用中遇到的基本问题，即如何在 P2P 网络中找到存有特定数据的结点。Chord 专门为 P2P 应用设计，因此考虑了在 P2P 应用中可能遇到的特殊问题，在 P2P 系统中处于较重要的位置。

1．哈希算法与 SHA-1 函数

Chord 使用一致性哈希作为哈希算法。在一致性哈希协议中并没有定义具体的算法，在 Chord 协议中将其规定为 SHA-1。

SHA，全称为 Secure Hash Algorithm，译作安全散列算法，是美国国家安全局（NSA）设计的，由美国国家标准与技术研究院（NIST）发布的一系列密码散列函数。正式名称为 SHA 的家族第一个成员发布于 1993 年。

1993 年 SHA 家族的第一个成员 SHA-0 正式发布，之所以称为 SHA-0，是因为随着 SHA 的不断发展与改进，人们通过这非正式的名称 SHA-0 以避免与它的后继者混淆。两年之后，SHA-1 出现，也就是第一个 SHA 的后继者发布了。随着应用的扩展和变化，SHA 系列又有其他的改进用以提升输出的范围和变更一些细微设计，这又产生了另外 4 种变体：SHA-224、SHA-256、SHA-384 和 SHA-512（这些有时候也被称作 SHA-2）。这就是 SHA-1 函数的由来。

2．Chord 建立拓扑的方式

Chord 建立拓扑的方式主要是以下几个过程：

将每一个加入结点 IP 地址利用哈希函数（Hashing function），产生一个结点代码（Node ID），这个代码会对应到一个由 N 个整数所形成的逻辑代码环（identifier circle），代表结点的位置，如图 2.17 所示。有 N1、N8、…、N51 等结点，即是透过哈希函数产生代码，并对应到逻辑代码环上的位置概念图。

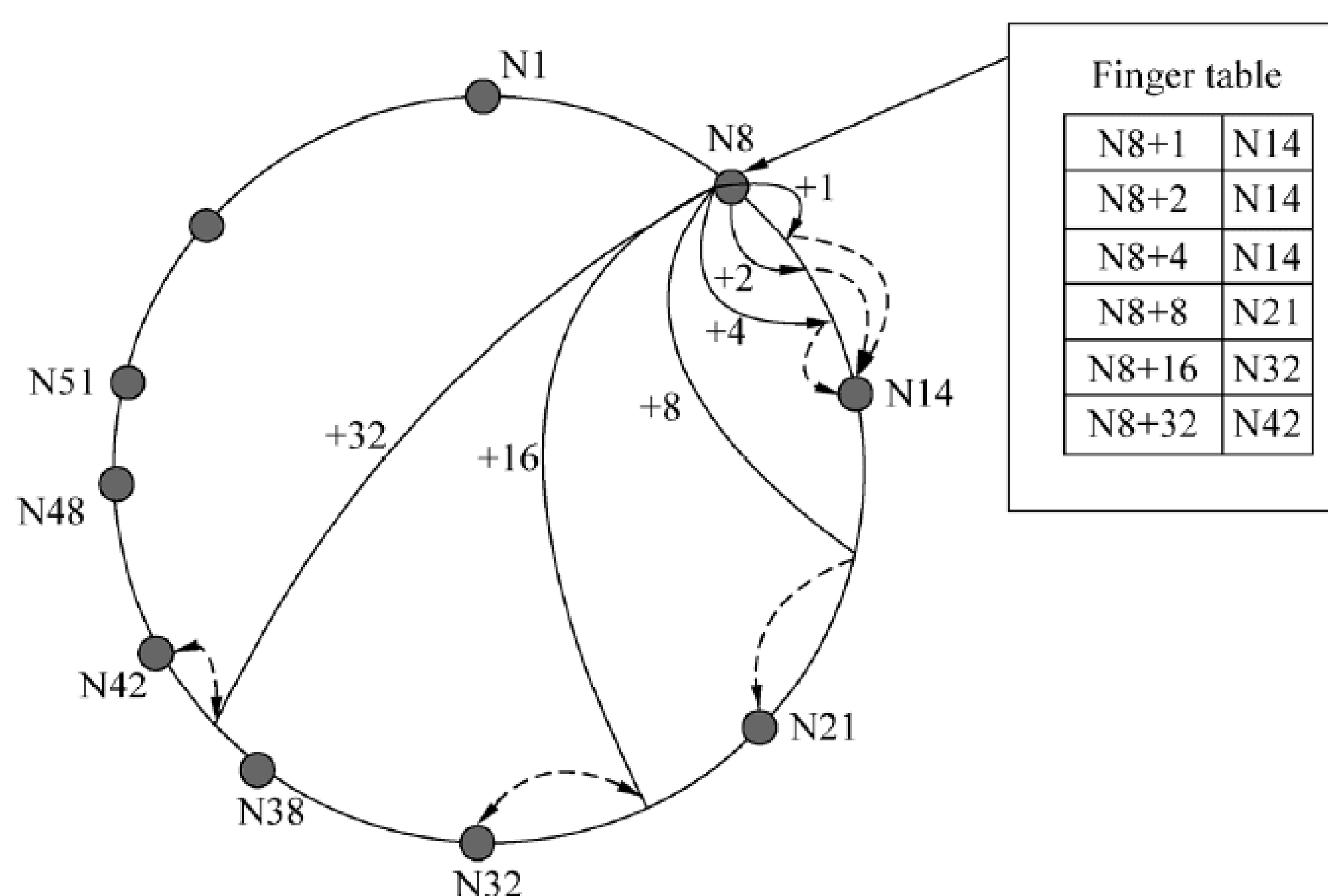


图 2.17　算法原理图

当所加入的结点将资源分享到 Chord 的系统时，Chord 也是利用哈希函数将所分享的资源产生代码，对应到环状代码环上的结点储存起来。例如代码为 1 的资源，则储存在结点代码 1 中，代码为 2～8 的资源则储存在结点代码 8 中，依次类推。

为了加快资源搜索的速度，每个结点都会维护一个路由表（Finger table），路由表内所记录的为每个结点的次结点（successor）位置。所谓次结点就是沿着逻辑代码环顺时针方向前进所遇到的第一个结点，记录次结点的内容为（n+2i−1），n 为该结点的代码，1≤i≤logN。

当结点搜寻资源时，则利用代码来比对结点中的路由表，进行绕送。

以图 2.17 为例来说明在 Chord 中资源的搜寻方式，当结点代码 8 要搜寻资源代码 42

时，结点代码 8 则透过结点中的路由表，开始寻找资源存在的位置。

在图 2.17 的绕送表中，可以知道到 N8+32 所对应的结点代码为 N42，因此，透过结点的绕送表查询，可以直接跳跃到所查询到的结点上，而不需要依照顺时针方向依序询问代码环上的结点，达到加快搜索的速度。

3．Chord 算法优点

Chord 算法与其他算法相比有一定的优越性，这些优点主要体现在：

- 负载平衡：这一优点来自于一致性哈希，也就是一致性哈希中提到的平衡性。所有的结点以同等的概率分担系统负荷，从而可以避免某些结点负载过大的情况。
- 分布性：Chord 是纯分布式系统，结点之间完全平等并完成同样的工作。这使得 Chord 具有很高的鲁棒性，可以抵御 DdoS 攻击。
- 可扩展性：Chord 协议的开销随着系统规模（结点总数 N）的增加而按照 O(log N) 的比例增加。因此 Chord 可以用于大规模的系统。
- 可用性：Chord 协议要求结点根据网络的变化动态地更新查询表，因此能够及时恢复路由关系，使得查询可以可靠地进行。
- 命名的灵活性：Chord 并未限制查询内容的结构，因此应用层可以灵活地将内容映射到键值空间而不受协议的限制。

2.5.7　Tapestry 项目

Tapestry 是由加州柏克莱大学所提出的点对点搜索架构，提供了一个分布式容错查找和路由基础平台，在此平台基础之上，可以开发各种 P2P 应用，其思想来源于 Plaxton。在 Plaxton 中，结点使用自己所知道的邻近结点表，按照目的 ID 来逐步传递消息，而 Tapestry 基于 Plaxton 的思想，加入了容错机制，从而可适应 P2P 动态变化的特点。

Tapestry 架构中，先由哈希函数将结点的 IP 与资源转成代码，所不同的是其代码由 16 进位的数字所组成。Tapestry 点对点系统架构的结点加入与搜索资源的方式，主要都是利用结点代码之间字尾比对的方式来形成一个网络拓扑。

当新结点加入时，新结点必须先通过系统所默认的网关结点（Gateway node）取得其路由表最接近的结点位置信息，再通过最接近结点内的路由表与新加入的结点代码做字尾比对，找出最接近新加入结点的代码。以此类推直到找到与新加入结点最接近的结点，与其连接并取得其结点所负责管理的资源地址数据。其主要的搜索方式与结点的布建方式都与前述的算法大同小异。

注意：最近，Tapestry 为适应 P2P 网络的动态特性，做了很多改进，增加了额外的机制实现了网络的软状态（soft state），并提供了自组织、鲁棒性、可扩展性和动态适应性。当网络高负载且有失效结点时，性能有限降低，消除了对全局信息的依赖、根结点易失效和弹性差的问题。

2.5.8　Pastry 项目

Pastry 是微软研究院提出的可扩展的分布式对象定位和路由协议，可用于构建大规模

的 P2P 系统。

Pastry 点对点系统与 Chord 点对点系统都是利用环状逻辑拓扑架构来设计的点对点网络架构。在 Pastry 中，每个结点分配一个 128 位的结点标识符号（nodeID），所有的结点标识符形成了一个环形的 nodeID 空间，范围从 0～2^{128} －1，结点加入系统时通过散列结点 IP 地址在 128 位 nodeID 空间中随机分配。如图 2.18 所示为 Pastry 的消息路由示意图。

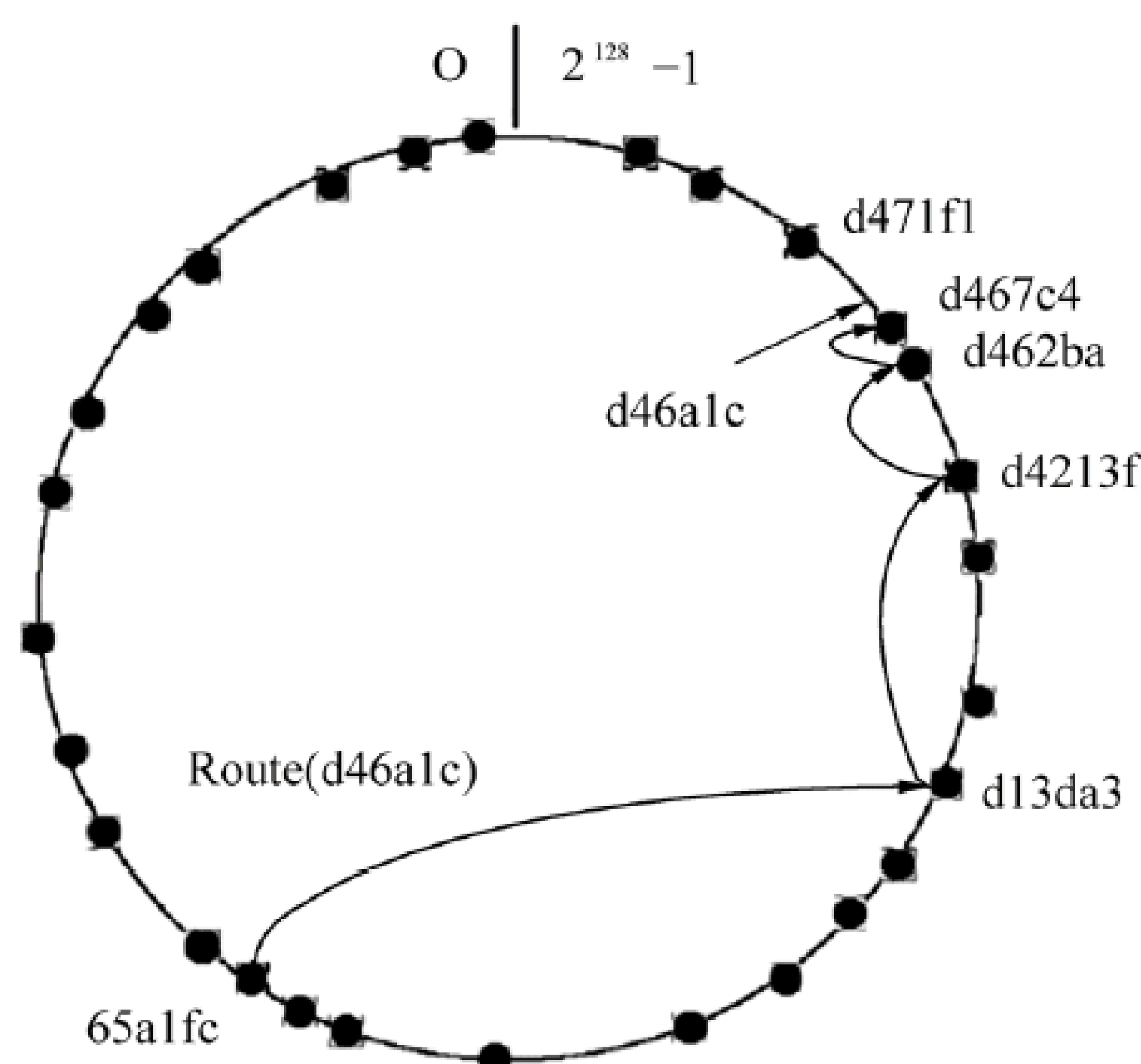


图 2.18　Pastry 的消息路由

Pastry 系统的工作原理主要是利用两个不同的哈希函数，将结点 IP 与文件转换成 128b 的结点代码与文件代码，而每个结点也有路由表（routing table）来记录结点的代码与文件代码间对应关系的数据。为了达到快速搜索的目的，Pastry 系统结点的路由表将所记录的结点分为 Leaf set、Routing table 及 Neighborhood set 这 3 个部分，分别为：

- Leaf set 所记录的为与结点所相邻的结点代码；
- Neighborhood set 所储存的信息为相近的结点代码。当需要搜寻档案时，每个结点会先搜索比对是否在 Leaf set 或是 Neighborhood 内；
- Routing table 所记录的信息为与结点代码有相同前置码（prefix）的结点。

当 Pastry 在进行资源搜索时，会先判断所要寻找的结点是否在 Leaf set 或是 Neighborhood set 中。如果存在则可以马上找到资源存在的结点。如果没有在 Leaf set 或是 Neighborhood set 中，则通过 Routing table 中的前置码，找出最接近的结点，再从所找出最接近的结点中继续搜索。

注意： 读者在学习过程中，这些理论性的知识可能较难理解，难以理解的地方暂可不必深究。这里讲述的只作简要了解即可，在后面章节中，关于 P2P 网络的搜索技术里还会讲到这些。P2P 网络的拓扑和搜索技术是 P2P 技术的核心，整本书都会反复地再现这些知识点，读者要着重理解并掌握。

2.5.9　全分布式结构化 P2P 拓扑存在的问题

上述几种分布式的 P2P 系统都是基于 DHT 的，而 DHT 这类结构最大的问题是 DHT

的维护机制较为复杂，尤其是结点频繁加入和退出所造成的网络波动（Churn）会极大增加 DHT 的维护代价，这是全分布式结构化的 P2P 拓扑存在的主要问题。

DHT 所面临的另外一个问题是 DHT 仅支持精确关键词匹配查询，无法支持内容/语义等复杂查询。在网络技术快速发展的今天，智能搜索、语义查询等技术成为必然的趋势，P2P 网络及其搜索技术在这方面必须要有所突破。

2.6　全分布式非结构化的 P2P 网络

上文已经讲述了全分布式结构化的 P2P 网络，在这类网络中，对信息定位有严格限制，也正因为此才称之为结构化（structured）的 P2P 系统。在全分布式的结构中，还有另一类 P2P 系统，这类系统对信息定位没有严格限制，信息自由存储，通常称这类系统为非结构化（Unstructured）的 P2P 系统。本节重点讲述在全分布式的条件下非结构化的 P2P 网络拓扑结构。

2.6.1　非结构化 P2P 网络拓扑概述

全分布式非结构化的 P2P 网络系统构建比较简单随意，在实际应用中适合于信息发布、即时通信等主机随时加入和退出的情况。

目前，大多数的 P2P 应用系统是非结构化拓扑结构的，这种结构的覆盖网络一般采用基于完全随机图的组织方式，结点度数服从 Power-law 规律（幂次法则），从而能够较快发现目的结点。

采用非结构化的 P2P 系统，对网络的动态变化有较好的容错能力，具有较好的可用性。同时可以支持复杂查询，如带有规则表达式的多关键词查询、模糊查询等，采用这种拓扑结构最典型的案例便是 Gnutella（音译：纽特拉）。目前基于 Gnutella 网络的客户端软件非常多，著名的有 Shareaza、LimeWire 和 BearShare 等。下面就以实例来讲解一下非结构化 P2P 系统的相关知识。

注意：幂次法则指的是个体的规模和其名次之间存在着幂次方的反比关系，R(x)=ax-b。其中，x 为规模（如：人口、成绩、营业额...），R(x)为其名次（第 1 名的规模最大），a 为系数，b 为幂次。这一现象在 100 多年前即被发现。许多的经验研究发现，诸如都市人口、网站规模、（英文）字汇出现频率、国民生产总值等，均呈现幂次法则现象。读者如有兴趣可自行参考相关资料。

2.6.2　全分布式非结构化 P2P 网络的应用实例

Gnutella 网络的第一个客户端由 Nullsoft 公司的贾斯汀·法兰科（Justin Frankel）与汤姆·帕勃（Tom Pepper）于 2000 年早期最先开发。同年 3 月 14 日，该程序被放在 Nullsoft 的服务器上并允许公众下载。该程序的源代码在 GNU 通用公共许可证下被发布。

Gnutella 是全分布式非结构化 P2P 系统应用的经典实例，它是一个 P2P 文件共享系统。

准确地说，Gnutella 不是特指某一款软件，而是指遵守 Gnutella 协议的网络以及客户端软件的统称。在后文中会有对 Gnutella 协议的详细分析。

Gnutella 和 Napster 最大的区别在于 Gnutella 是纯粹的 P2P 系统，没有索引服务器，它采用了基于完全随机图的洪泛（Flooding）发现和随机转发（Random Walker）机制。为了控制搜索消息的传输，通过 TTL（Time To Live）的减值来实现。

注意： TTL，生存时间的意思，用来指定数据包被路由器丢弃之前允许通过的网段数量，它是 IP 协议包中的一个值。在具体的执行过程中，用这个值来告诉网络路由器包在网络中的时间是否太长而应被丢弃。

如图 2.19 是 Gnutella 的网络结构模型，图中展示了查询文件的一次 Flooding 工作流程，当一台计算机要下载一个文件时：

（1）它首先以文件名或者关键字生成一个查询；

（2）把这个查询发送给与它相连的所有计算机；

（3）这些计算机如果有这个文件，则与查询的机器建立连接，如果没有这个文件，则继续在自己相邻的计算机之间转发这个查询；

（4）重复以上过程，直到找到文件为止。

为了控制搜索消息不至于永远这样传递下去，通过设置 TTL 来控制查询的深度。

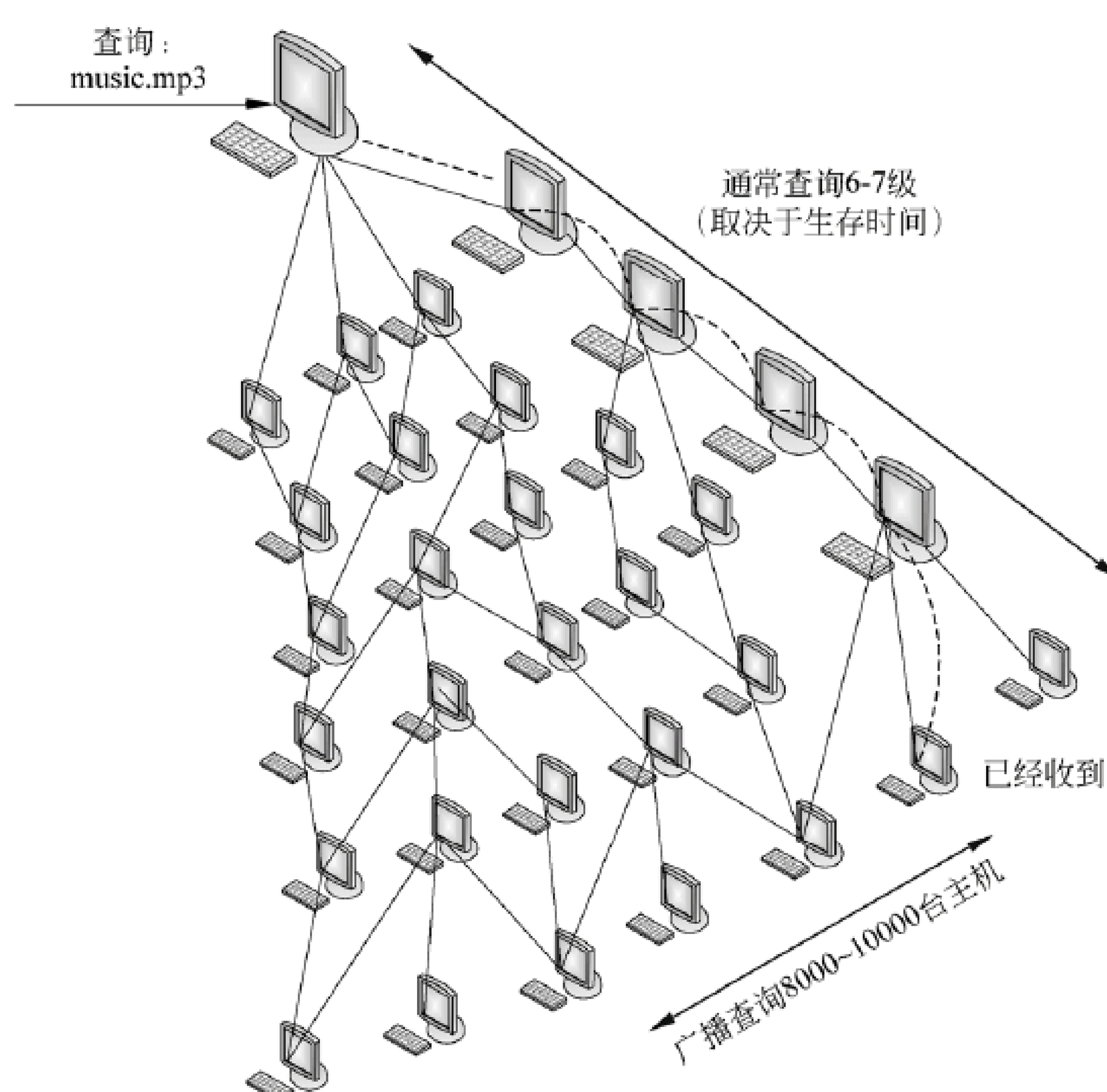


图 2.19　全分布式 Gnutella 网络模型结构图

在实际应用中，这种在 Gnutella 网络中的搜索模式是十分不可靠的。由于每一个结点都是一台普通的计算器用户，他们经常连接或者断开网络，所以整个 Gnutella 网络结构永远都不是完全稳定的。Gnutella 网络搜索的带宽消耗随着连接用户的增加而指数递增，经常饱和的连接会导致较慢的结点失去作用。因此，搜索请求在网络中会被经常丢弃，与整个网络相比，大多数的查询只会到达其中的很少一部分结点。

因为这些问题，在初期的 Gnutella 网络中，存在比较严重的分区，断链现象。也就是

说，一个查询访问只能在网络的很小一部分进行，因此网络的可扩展性不好。所以，后来许多研究人员在 Flooding 的基础上作了许多改进，例如采用 Random work、Dynamic Query 等方法。

2.6.3　非结构化 P2P 网络存在的问题

在结构化的 P2P 网络中，随着联网结点的不断增多，网络规模不断扩大，通过这种 Flooding 方式定位对等点的方法将造成网络流量急剧增加，从而导致网络中部分低带宽结点因网络资源过载而失效。

由于没有确定拓扑结构的支持，分布式的 P2P 网络无法保证资源发现的效率，因此发现的准确性和可扩展性是非结构化网络面临的两个重要问题。目前对此类结构的研究主要集中于改进发现算法和复制策略以提高发现的准确率和性能。

全分布式 P2P 网络结构，包括结构化和非结构化的，它们都较好地解决了网络结构中心化的问题，扩展性和容错性较好，通常也称为第二代 P2P 结构，是对第一代集中式 P2P 网络结构的优化和完善，在 P2P 的研究和发展过程中，有重要的地位。

2.7　混合式 P2P 网络结构

随着 P2P 技术的不断发展和实际应用的检验，第一代 P2P 和第二代 P2P 的优点在应用中都得到了集中的体现，与此同时它们也都暴露了不少难以克服的问题。在这种背景下，研究者们将分布式 P2P 去中心化和集中式 P2P 快速查找的优势综合起来，便形成了一种混合式 P2P 网络结构，也叫半分布式的 P2P 结构。这种结构是分布式 P2P 和集中式 P2P 二者优点的有机结合，通常也叫第三代 P2P 网络结构，本节重点讲述这种 P2P 拓扑结构的相关知识。

2.7.1　混合式 P2P 网络的原理

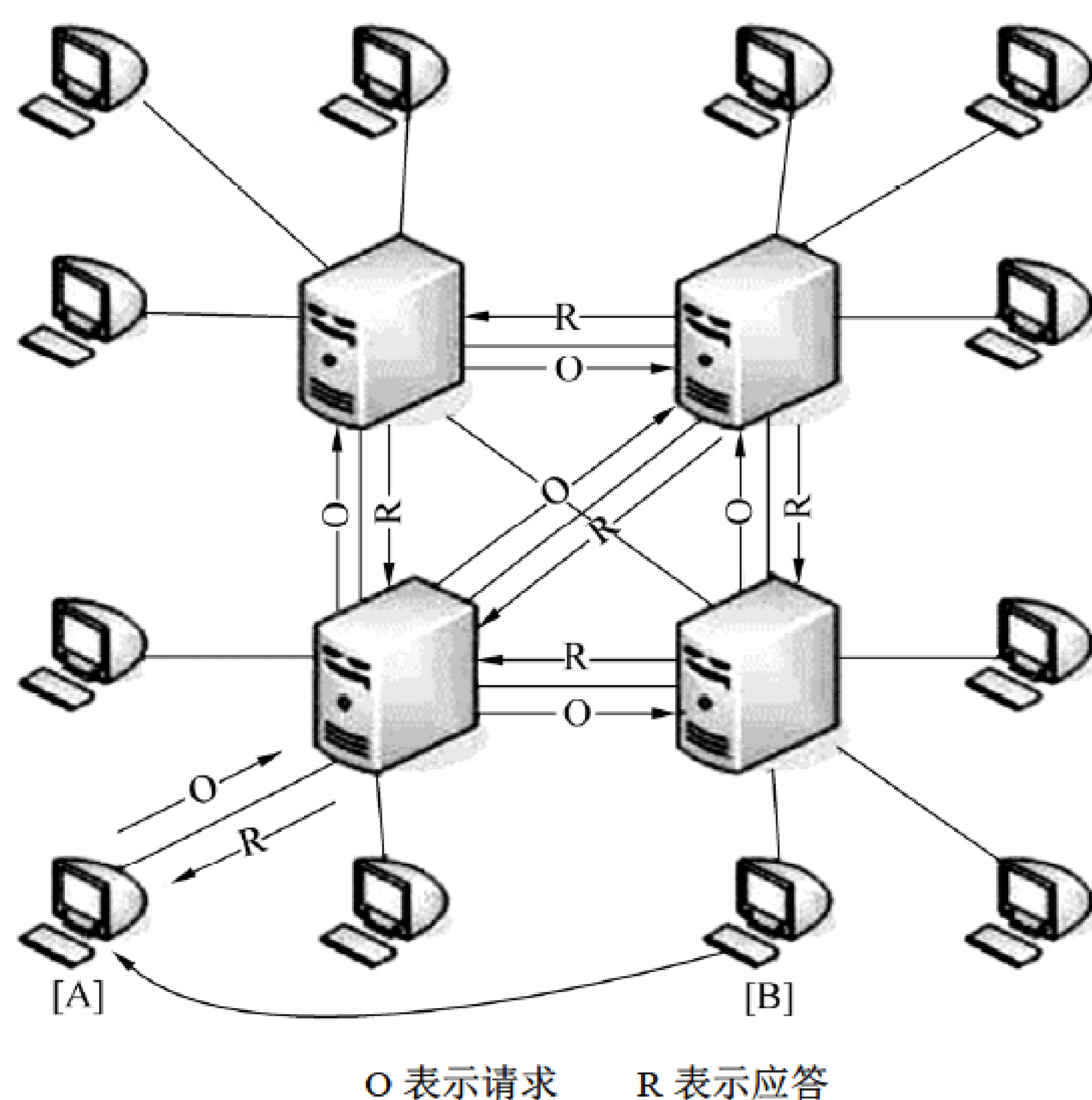
在混合式 P2P 网络结构中，将整个网络中的结点按能力不同（计算能力、内存大小、连接带宽、网络滞留时间等）区分为普通结点和超级结点两类。超级结点也叫搜索结点，它与其临近的若干普通结点之间构成一个小型的、自治的、基于集中式的 P2P 网络模式，如图 2.20 所示为混合式 P2P 网络结构模型。

在图 2.20 所示的结构模型中，处于中间位置的 4 台大主机定义为超级结点，类似于一个小型集中式 P2P 网络的目录服务器。整个 P2P 网络中再通过分布式的 P2P 网络模式将超级结点相连起来，就组成了一个混合式的网络结构。

混合式 P2P 网络拓扑中包含两类结点，一类是搜索结点，另一类是普通结点。搜索结点与其临近的若干普通结点之间构成一个自治的“簇”，在每一个“簇”内采用基于集中目录式的 P2P 模式。整个 P2P 网络中各个不同的“簇”之间再通过纯 P2P 的模式将搜索结点相连起来，这样就组成了一个混合式的拓扑。

这种混合式的 P2P 拓扑结构在工作过程中，一般会选择一些性能较好的结点作为超级

结点，也就是索引结点。当有结点加入或退出时，系统可以在各个搜索结点之间再次选取性能最优的结点，或者另外引入一新的性能最优的结点作为索引结点来保存整个网络中可以利用的搜索结点信息，并且负责维护整个网络的结构。



O 表示请求　　R 表示应答

图 2.20　混合式 P2P 网络模型结构图

2.7.2　混合式 P2P 网络的优缺点

在混合式的 P2P 网络拓扑中，由于在各个超级点上存储了系统中其他部分结点的信息，发现算法仅在超级点之间转发。如果查询结果不充分，超级点再将查询请求转发给适当的叶子结点进行有限的泛洪。这样就极为有效地消除了纯的分布式 P2P 结构中使用泛洪算法带来的网络拥塞、搜索迟缓等不利影响。

另一方面，每个簇中的搜索结点负责监控所有普通结点的行为，能确保一些恶意的攻击行为在网络中得到局部控制，在一定程度上提高整个网络的负载平衡。

但这种结构也有不足，因为其对超级点依赖性大，易于受到集中攻击，容错性也受到影响。

混合式 P2P 网络结合了集中式拓扑的易管理性与分布式拓扑的可扩展性，在异构的 P2P 网络环境下是一种较好的模式选择。其中最典型的案例就是 KaZaa。

注意：在混合 P2P 网络结构中，当网络规模扩大时，就可以在各个超级结点之间再次选取性能最优的结点，或者另外引入一新的性能最优的结点作为超级结点，这样超级结点能随着网络规模扩大而自适应的增加。

2.7.3　混合式 P2P 拓扑的应用实例——KaZaa

混合式 P2P 网络拓扑最典型的应用就是 KaZaa，KaZaa 是现在全世界流行的几款 P2P

软件之一，可以使用 KaZaa 下载会员们分享出来的资源。使用 KaZaa 可以实现简易的搜寻，也可快速地找到想要的文件，包括音乐、影片、软件、游戏、图片等。在新版本的 KaZaa 中还支持防火墙、更快速的查询及 BUG 修正等。如图 2.21 是 KaZaa 的系统界面图。

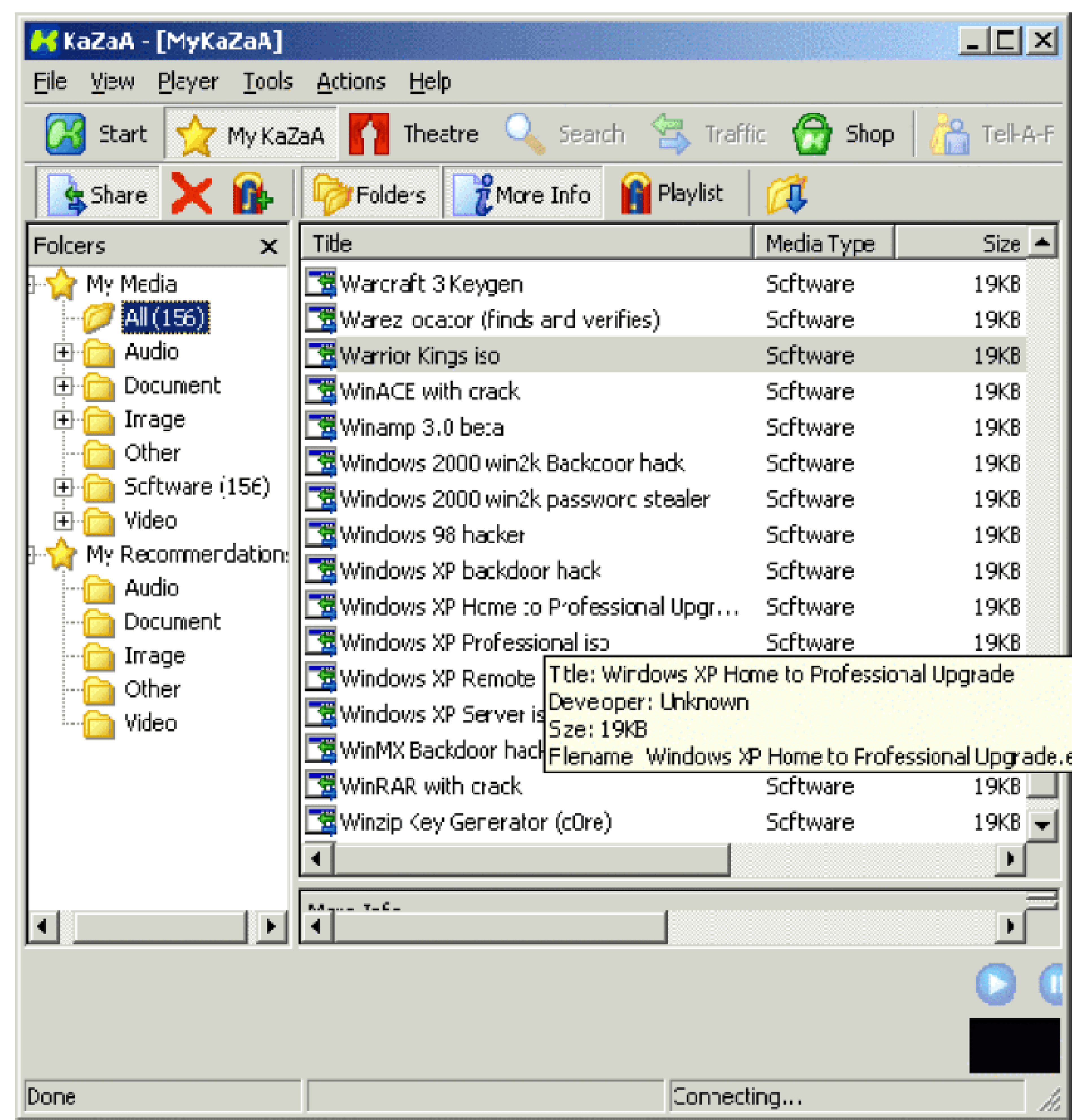


图 2.21　KaZaa 系统运行界面图

KaZaa 在当前 P2P 的应用系统中是比较流行的一个，根据 CA 公司统计，全球 KaZaa 的下载量超过 2.5 亿次。使用 KaZaa 软件进行文件传输消耗了互联网 40%的带宽。

KaZaa 之所以如此成功，是因为它结合了 Napster 和 Gnutella 共同的优点，这正是混合式 P2P 网络所具备的特点。从结构上来说，它使用了 Gnutella 的全分布式的结构，这样可以使系统得到更好的扩展，因为它无须中央索引服务器存储文件名，是自动地把性能好的机器称为 Super Node（超级结点）。它存储着离它最近的叶子结点的文件信息，这些 Super Node，再连通起来形成一个 Overlay Network（覆盖网），由于 Super Node 的索引功能，使搜索效率大大提高。

2.7.4　P2P 网络拓扑结构的集中对比

上文中已经对 P2P 网络的常用 4 种拓扑结构进行了详细的讲解，不同的结构不管是在理论研究还是实际应用中都有不同的特点。如表 2.1 从可扩展性、可靠性、可维护性、发现算法的效率、复杂查询等方面比较了这 4 种拓扑结构的综合性能。

表 2.1　P2P网络结构综合性能对比表

比较标准 / 拓扑结构	中心化拓扑	全分布式非结构化拓扑	全分布式结构化拓扑	半分布式拓扑
可扩展性	差	差	好	中
可靠性	差	好	好	中
可维护性	最好	最好	好	中
发现算法的效率	最高	中	高	中
复杂查询	支持	支持	不支持	支持

2.8　发展中的 P2P 技术

发展中的 P2P 技术也称为第四代 P2P 技术，之所以称为发展中的 P2P 技术是因为这种技术正处在研究、论证的阶段，并没有达到可以统领一代的技术水准。在这一类 P2P 技术中，大部分都是在原有技术的基础上进行改进，根据现有技术的缺陷和不足提出并应用了一些新技术措施。综合这些技术，典型的有以下几种：

（1）动态端口选择：目前的 P2P 应用一般使用固定的端口，但是一些公司已经开始引入协议可以动态选择传输端口，一般来说，端口号在 1024～4000 之间。甚至 P2P 流可以使用原来用于 HTTP（SMTP）的端口 80（25）来传输以便隐藏。这将难以识别来自哪个运营商网络的 P2P 流，掌握并测量其流量将变得更困难。

（2）双向下载：eD 和 BT 等公司进一步发展引入双向流下载。该项技术可以多路并行下载和上传一个文件和/或多路并行下载一个文件的一部分。目前传统的体系结构要求目标在完全下载后才能开始上传，这将大大加快文件分发速度。

注意： 这里的 eD 是 eDonkey 的意思，与常说的 eMule 类似，在本书的第 7 章会有这两个概念的详细说明。

（3）智能结点弹性重叠网络：弹性重叠网（Resilient Overlay Networks，简称 RON），作为应用层被重叠在现有的互联网选路层上。RON 的结点监视互联网路径的机能和质量，根据这些信息决定是直接利用它来传输数据还是通过其他 RON 的结点。用 RON 可以减少丢包率，降低时延，提高吞吐量。在几秒钟内发现路径损耗和周期性性能下降并且使之恢复正常。RON 作为一种新的 P2P 技术方案，它的设计目标主要有 3 个：RON 的第 1 个目标是通过主动探测和监视结点之间的链路来发现问题；第 2 个目标是将选路和路径选择与分布应用综合，使之具有根据应用特性来选择路径的能力；第 3 个目标是提供框架，以实现管制路径选择的显式路由策略，用来管理网络中路径的选择。

除了以上这些，还有很多其他的 P2P 技术都在不断的研究和发展中，所有这些技术其根本目的都是最大限度地解决 P2P 存在的问题，以促进 P2P 技术的纵深发展。

2.9　本 章 小 结

在 P2P 系统中，设计者可以根据不同的应用需求组建不同拓扑结构的对等网络（P2P 网络）。一个“合适”的网络拓扑可以为应用提供更好的支持，例如规则的拓扑结构有利于根据主机在拓扑网络中的位置进行基于内容的索引。非结构随机网络的构建比较简单随意，所以适合于信息发布、即时通信等主机随时加入退出的情况。因此理解对等网络的拓扑对 P2P 系统的研究有重要的意义。

本章正是基于这一点出发，重点讲解了 P2P 网络拓扑相关的技术，从网络拓扑的基本概念到 P2P 网络拓扑的分类及原理都作了讲解。学习 P2P 网络拓扑知识是理解 P2P 的基础，也是学习和研究 P2P 必备的知识。

在学习本章的过程中，有较多的理论知识需要理解，也涉及不少算法和数据结构的知识，如有不明白的地方可先行参考相关知识点。

第 3 章　P2P 网络的搜索技术

第 2 章主要讲述了 P2P 网络体系结构的相关知识，P2P 网络的体系结构对 P2P 网络的性能和功用有着十分重要的作用。然而，一个完善的 P2P 网络系统的成功与否不仅仅在于其网络结构的合理和有效，在很大程度上还取决于其资源查找机制的灵活性和可扩展性。在 P2P 网络中，资源的查找其实质就是 Peer 结点的搜索与发现，这种搜索技术是 P2P 技术体系中一个非常重要的方面。

P2P 网络中存储着大量的资源，要想充分利用 P2P 网络上的资源，就必须要有一个良好的资源发现机制，即能在 P2P 网络中对各类资源信息进行高效的搜索。目前，P2P 研究中的一个主要问题就是 P2P 网络中的资源搜索问题。

由于 P2P 系统的可扩展性和网络结点的不确定性，设计一个良好的搜索机制比较困难。因此，高效的搜索机制是 P2P 研究的关键技术之一，并且一直是网络研究最活跃的领域之一。针对 P2P 技术在搜索领域的关键性问题，本章将从 P2P 搜索技术涉及的各方面来进一步研究 P2P 技术。本章知识点如下。

- 搜索与搜索引擎：要理解什么是搜索、搜索引擎等知识，对搜索这一概念要有一个全局性的理解。
- 什么是 P2P 搜索：要明白 P2P 搜索具体是做什么用的，在 P2P 网络中为什么需要搜索。
- 理解 Web 搜索与 P2P 搜索的异同：P2P 搜索与我们通常使用的 Web 搜索在基本原理、实现机制上都是不同的，重点要理解有哪些地方不同。
- P2P 搜索技术的评价标准：要了解什么样的搜索算法和搜索实现策略才是好的、高效的、可用的，在设计 P2P 搜索算法时的依据是什么。
- P2P 搜索技术的内容：P2P 搜索技术包括集中式 P2P 搜索、结构化的 P2P 搜索、非结构化的 P2P 搜索及其他的混合 P2P 搜索技术，这些都是学习 P2P 搜索技术必须掌握的内容。
- P2P 搜索技术面临的挑战和未来发展：简要了解当前 P2P 搜索技术面临的问题及进一步的研究方向。

学习本章的重点是要在理解 P2P 网络拓扑结构的基础上，理解搜索算法的机制和实现原理。

3.1　搜索与搜索引擎

搜索，在日常生活中是一个再熟悉不过的概念，在使用互联网的时候，每时每刻都会用到与搜索有关的东西，或是执行与搜索有关的操作。互联网的信息浩如烟海，信息的组

织与存储分布在全球每个角落的计算机里，要想找到自己需要的资源，必需借助相应的搜索功能才能完成。尽管这种搜索的概念与将要讲到的 P2P 网络中的搜索有一定的区别，但理解搜索及搜索引擎的相关知识是学习本章的必要前提，本节主要讲搜索和搜索引擎的相关知识。

3.1.1　互联网的大搜索时代

中国互联网十多年的发展历史中，搜索真正启蒙的时间也就最近一两年。搜索从雅虎时代到 google 时代的更替仅仅花了几年的时间，业界有这样的评价，雅虎搜索的没落是因为其处于搜索与门户交替的时代，雅虎最后选择了门户，而 google 专注于搜索。中国互联网搜索大战，2005 年是关键年，注定了现今中文搜索市场的格局。百度坐上了中文搜索的头把交椅，谷歌中国也在锲而不舍地追赶，试图超越。而就在此时，微软也不甘寂寞，推出自己的搜索引擎 bing，至于其他的各类不同应用、不同风格的搜索引擎更比比皆是，如图 3.1 展示了当前互联网中多元化的搜索引擎，可以说，互联网搜索市场现在是群雄逐鹿。



图 3.1　众多的搜索引擎

目前，从全球来看，引领搜索技术发展的就是 Google，就当前搜索引擎的搜索模式来说，只是对海量的网页进行简单排序。事实上，最聪明的搜索引擎，并非要提供多元化信息，而是能够提供用户提出问题的答案，甚至解决问题。用户的要求不同，搜索结果也应该不同。换言之，搜索引擎的发展，其实刚刚开始。从搜索的发展来看，基于海量搜索之后，搜索引擎将越来越贴近为用户解决实际问题，提供精准搜索和专业化搜索。而分类搜索也将实现对用户信息的整合，整合搜索、社区搜索和移动搜索将成为今后搜索引擎的发展趋势。

中国约有 13 亿人口，搜索在中国是一个巨大的市场。2008 年中国网民约有 2.53 亿，域名注册量居世界第一，宽带用户超美国。随着历史车轮的向前迈进，网民将越来越多，

网络需求势必在质量以及市场上再一次细分。全国 265 个主要城市，涵盖衣食住行、吃喝玩乐、教育医疗、家政便利、市政设施等 110 个类别，网民所需的信息是庞大而全面的。越来越多的网民习惯于在网上搜索自己想要的东西。

当前的互联网是一个基于搜索应用的互联网，互联网的大搜索时代来临，在这个发展的大潮流下，P2P 也不例外。

3.1.2　Web 搜索引擎

互联网上的搜索活动是建立在某一搜索引擎基础上的。搜索引擎，简单地说就是一种基于 Internet 的信息查询工具，包括信息的存储、整理和检索 3 个过程，如百度、Google 等，就是典型的搜索引擎实例。

从应用的角度来讲，搜索引擎就是一种工具，是一种可以搜集互联网上几千万到几十亿个网页，并对网页中包含的每一个关键词进行索引并建立索引数据库的交互式全文查询工具。根据搜索获取信息方式的不同，通常搜索引擎可以分成以下 3 种。搜索引擎的分类如图 3.2 所示。

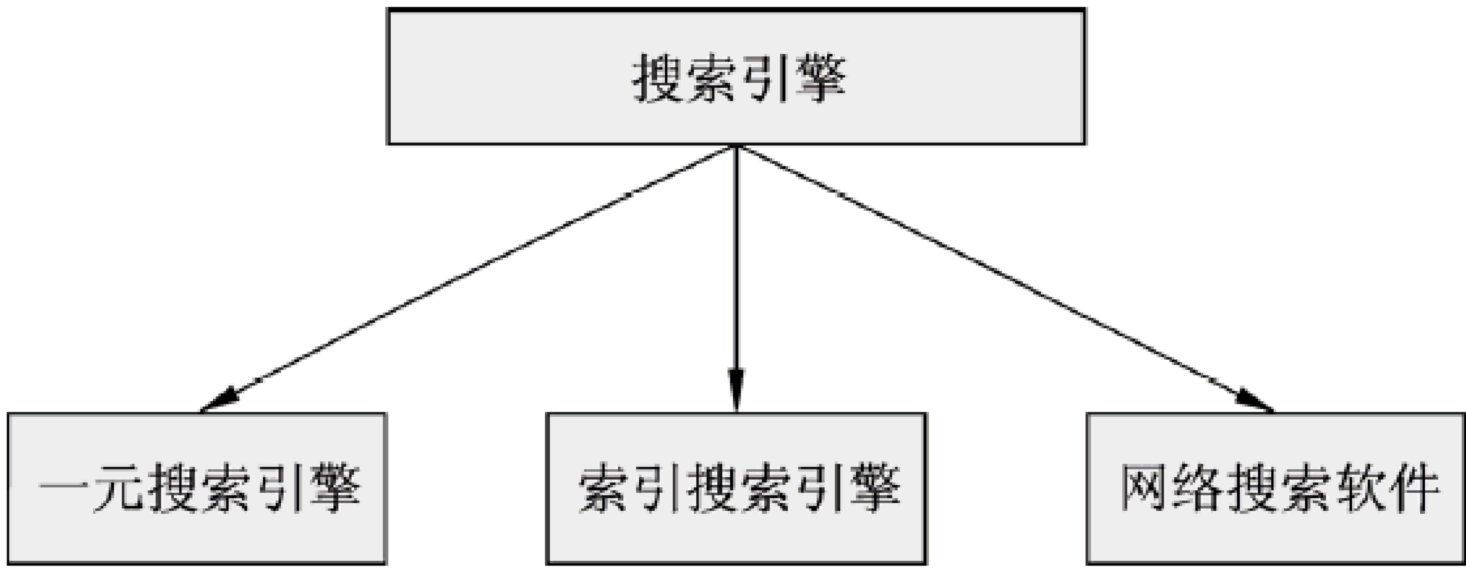


图 3.2　当前互联网中搜索引擎的分类

- 一元搜索引擎：利用 Robert、Spider 或 Crawlers 等搜索软件，沿 WWW 网站的链接自动漫游，不断搜集网上的各种信息资源，形成包含成千上万记录的索引数据库。当用户输入检索请求时，搜索软件将关键词与索引数据库中的词相匹配，自动生成结果返回用户。
- 索引搜索引擎：这种搜索引擎一般没有自己的数据库，对于用户的检索请求，通常通过转义在其他的一个或多个搜索引擎的索引数据库中搜索，对结果进行整理（查重、合并、校验、排序）后，返回给用户。
- 网络搜索软件：一般直接安装在本地计算机上，通过向多个搜索引擎发出检索请求来查询信息，对返回的结果根据一定的规则显示输出给用户。检索方法的特点类似于索引搜索引擎，但专用性及灵活性强过索引搜索引擎。

3.1.3　搜索引擎的工作原理

搜索引擎是一个提供信息检索和索引服务的网站，它使用某些程序把因特网上的所有信息归类以帮助人们在茫茫网海中搜寻到所需要的信息。搜索引擎执行一次搜索，要完成对信息的搜索并将结果展现出来，一般要完成下面 3 个方面的工作。

1．在因特网上抓取网页

专门用于检索信息的机器人程序如蜘蛛程序在网络间爬来爬去，自动收集、访问网页，并沿着网页中的所有 URL 爬到其他网页，蜘蛛程序不断重复这一过程，把到过的所有网页收集回来。随着因特网呈几何级数的迅速发展，使得检索所有新出现的网页变得越来越困难，传统的蜘蛛程序工作原理发生了一些改变，既然所有网页都可能有连向其他网站的链接，那么从一个网站开始，跟踪所有网页上的所有链接，就有可能检索整个因特网。

2．建立索引数据库

由分析索引系统程序对收集回来的网页进行分析，提取相关网页信息，包括网页所在 URL、编码类型、页面内容包含的所有关键词、关键词位置、生成时间、大小、与其他网页的链接关系等。根据一定的相关度算法进行大量复杂计算，得到每一个网页针对页面文字中及超链中每一个关键词的相关度或重要性，然后用这些相关信息建立网页索引数据库。

3．在索引数据库中搜索排序

当用户输入搜索关键词后，由搜索系统程序从网页索引数据库中找到符合该关键词的所有相关网页。因为所有相关网页针对该关键词的相关度是已经计算好的，所以只需按照现成的相关度数值排序，相关度越高，排名越靠前。最后，由页面生成系统将搜索结果的链接地址和页面内容、摘要等信息组织起来返回给用户。

以上搜索引擎的 3 个工作过程，如图 3.3 所示。

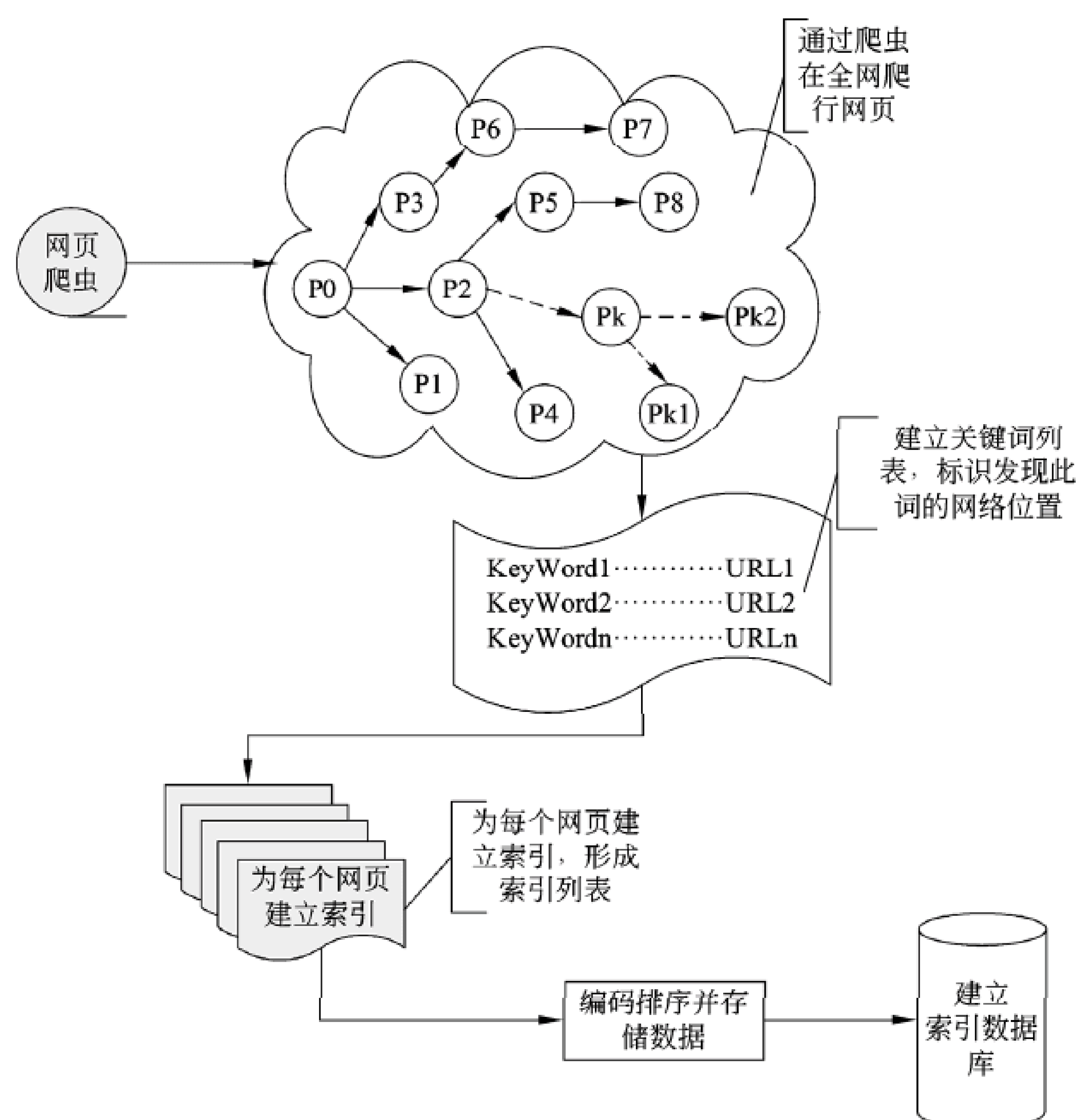


图 3.3　搜索引擎的工作原理

目前，在所有的搜索引擎中，英文搜索引擎中的 Google 和中文搜索引擎中的 Baidu（百度）是最具代表性的搜索引擎。它们的工作过程也会遵循如图 3.3 所示的几个步骤。

（1）首先由类似 Web spider 的网页爬行器在网络中不断的爬行，收集网页，在网络上获得海量的网页快照。

（2）然后分析这些网页并为每个网页建立索引，也就是图中的 Build Index 过程。

（3）最后将这些数据编码和排序规则存入索引数据库，供用户搜索访问。

实际上，当用户发起查询请求时，只是向数据库服务器发出了查询索引数据库的指令，然后数据库将查询结果按不同的归档方式返回给用户，用户就能得到查询结果了。

注意：这里讲的对 Google 或者 Baidu 搜索引擎工作步骤的说明，只是对通用搜索引擎的工作原理的简要描述，而并非是它们在搜索过程中实际的工作原理。搜索是一个复杂的过程，有兴趣的读者可自行查阅相关资料学习。

3.1.4　Web 搜索的优缺点

根据 Web 搜索引擎的工作原理，利用 Web 搜索存在一定的不足。

- Web 搜索结果与实际的数据信息之间存在滞后性；
- Web 搜索存在带宽瓶颈，对服务器的依赖程度太大导致系统过于脆弱。

注意：当在互联网中使用 Web 搜索引擎的时候，用户的查询请求能否得到响应，完全取决于服务器中是否存储有关于该关键字的网页索引。由于索引信息是由网络爬虫事先写进索引库中，这与真正的、当前的网页搜索存在着一定的滞后性。因此有可能发生查到网页的索引，却发现网页过期不可用的情况，在实际应用中也经常会出现这个问题。

当然，这种 Web 搜索技术也有它的优势，通常，Web 搜索的响应速度很快，一般的搜索用时都是毫秒级的，而且符合查询请求的结果很多，查询率高等。

3.2　P2P 搜索与 Web 搜索的异同

3.1 节简要介绍了搜索和搜索引擎的相关知识，明白了 Web 搜索的基本概念，下面就开始讲另一个搜索，就是 P2P 的搜索。P2P 研究领域一个重要的问题是搜索问题，但 P2P 搜索与常说的 Web 搜索是两个不同的概念，它们之间既有联系也有区别。为了能够更深入地理解 P2P 搜索，本节重点讲述一下 P2P 搜索及其与 Web 搜索的异同，学习 P2P 搜索技术先从了解 P2P 搜索开始。

3.2.1　什么是 P2P 搜索

P2P 网络的根本思想就在于对等和共享。在 P2P 系统中，资源是分散在各个结点之上的，并且结点频繁地加入或退出都相当自由，几乎没有规律可循。这些都使得 P2P 系统及

整个 P2P 系统的资源都处于不断的变化之中，那么如何找到并定位这些结点和资源，就是 P2P 需要解决一个重要问题。

P2P 要实现良好、高效、共享的机制，就要解决资源的搜索和结点的发现问题。所谓 P2P 搜索技术，就是一种 P2P 资源的发现和定位技术，通过搜索算法来发现、查找 P2P 网络中，在时间和空间上都处于动态的变化中的结点信息和资源存储信息，以最大限度、最快速度、尽可能多且准确的发现结点上的资源。这就是 P2P 搜索。

P2P 不但可以通过网络将所有电脑组建成一个对等的大型网络，还全面改写了当前的网络搜索技术。由于 P2P 软件特殊的工作方式，所以也就拥有了自己独有的搜索特性。在利用 P2P 软件搜索目标文件的时候不是像在传统的门户网站那样通过网站的服务器的数据库与所键入的关键字得出一些网站或者是底层页面的地址，而是将搜索的任务在所连接的计算机上一台一台地以接力的方式进行传输，直到找到自己所需的资源为止。可以说 P2P 网络中的搜索与通常意义上所说的 Web 搜索是两个不同的概念，它们的目的也许都是为了找到需要的资源，但是在实现机制、内部原理上却有着根本的不同，图 3.4 所示的就是一个基于 P2P 网络的简单搜索流程。

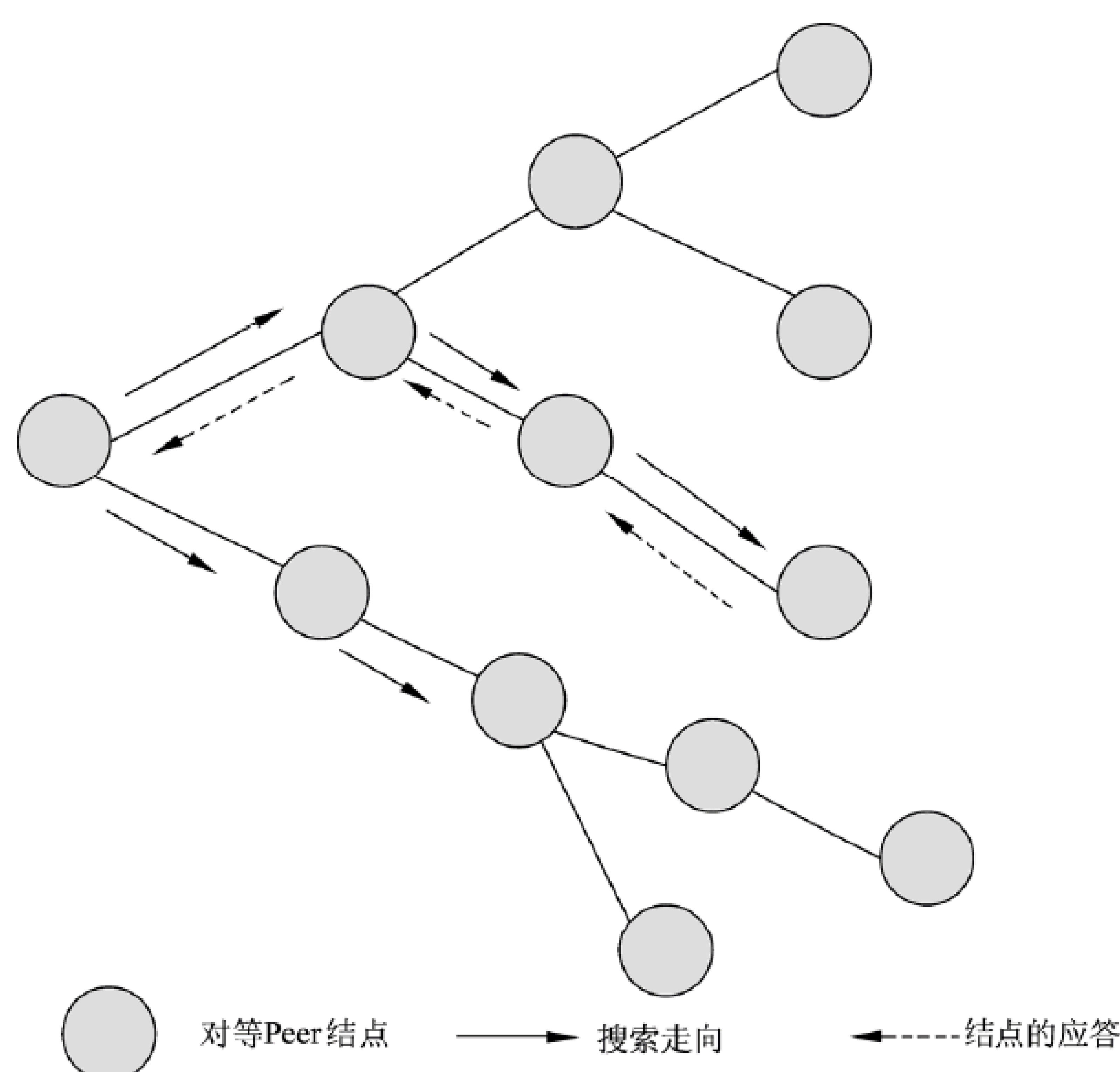


图 3.4　P2P 搜索的一般流程

从图 3.4 中可以看出，P2P 搜索的时候，查询信息在结点之间广播式的传递，扩散到整个网络空间，这与 Web 搜索是不相同的。通过和 Web 搜索引擎的对比，能够更深入地理解 P2P 的搜索原理和算法，以及 P2P 无可比拟的优越性。下面就简要的对比一下它们之间的异同。

3.2.2　面向的网络结构异同

P2P 本身是一个分布式的协作系统，因而 P2P 搜索所面向的网络环境也是一个架构在

Internet 上的分布式虚拟网络。而 Web 搜索则不同，它所执行的搜索是面向整个 Internet 网络的，也就是说，Web 搜索是基于全网的搜索。

在 P2P 搜索中，搜索算法及搜索的执行过程都是针对特定的 P2P 网络结构的，必须部署在 P2P 网络中才会起作用，它所应用的范围是在 P2P 网络的范围的，无法应用在任意的网络环境下。

所以总地来说，Web 搜索与 P2P 搜索所面向的网络结构是不相同的。Web 搜索是面向全网的，凡是加入到 Internet 中的主机结点理论上都适用于 Web 搜索。当然在 Web 搜索中还有一些特定需求，有些 Web 搜索是针对某一站点的。P2P 搜索则是面向 P2P 网络架构的。

3.2.3　搜索过程的异同

P2P 搜索与 Web 搜索的过程也不相同，在 P2P 网络中，参与 P2P 网络的用户将各自的资源共享出来，资源就存放在各个结点的 PC 上。当某一结点请求搜索的时候，它会把它的搜索请求通过 P2P 网络同时发给网络上另外 N 个与自己相邻的 PC 上，如果搜索请求未得到满足，这 N 台 PC 中的每一台都会把该搜索请求转发给另外与之相邻的 M 台 PC，这个过程不断进行下去，一直到找到所需资源或搜索过程终止。

图 3.4 所展示的搜索模式就是一个 P2P 搜索过程，其搜索范围将在几秒钟内以几何级数增长，几分钟内就可搜遍该 P2P 网络中几百万台 PC 上的信息资源。当然，这只是 P2P 网络中众多搜索形式中的一种。

而 Web 搜索的过程并非如此。当用户发出搜索命令后，Web 搜索引擎并不是即时的去互联网上搜索资源，而是去搜索预先整理好的网页索引数据库。它需要爬行网页、建立索引、入库等一步步地操作来相继完成。

图 3.3 是对 Web 搜索过程的形象描述，用户进行 Web 搜索的过程与 Web 搜索引擎的工作过程并不是同步的。搜索引擎事先将全网的资源搜索到并产生一个索引库，用户调用 Web 搜索引擎搜的过程，其实就是查询这个索引库过程。

3.2.4　资源发现策略的异同

P2P 资源发现的策略则是通过实时查询在线的 P2P 结点来得到响应。

P2P 的网络由多个自组织的结点所组成，每个连入 P2P 网络的结点都有自己的邻居结点。当某一个结点发起一个查询时，将这个查询的消息数据包广播到其邻居结点，或者根据某种哈希函数计算得到发送的目的结点。针对特定的查询请求，接收到查询消息的结点首先会判断自己是否符合查询的要求，然后决定是否按照一定的策略转发到其邻居中去。

因而，P2P 的搜索过程是实时的。当查询命中之时立刻可以通知发起者，并建立连接进行两个对等结点间的 P2P 交互任务。查询的命中与否不是取决于某个确定的结点，而是算法所规定的所有结点，因此，具有较好的鲁棒性。

Web 搜索并不像 P2P 搜索一步到位。用户发出的搜索命令并不是启动搜索引擎进行全网爬行，它对资源的发现策略一般都为 3 步，就是 3.1 节讲到的搜索引擎的 3 步工作原理，即先从互联网上抓取网页，然后建立索引数据库，最后其搜索过程就是在索引数据库中搜索资源。

通过对比可知，P2P 的搜索策略有着传统 Web 搜索无可比拟的深度，在理论上可以达到 100%。同时，在搜索文件时不会受到文件格式、存储形式、系统平台的限制，所有共享 P2P 网络中的相关文件甚至整个硬盘都能被检索并将结果呈现给发起者。这些都是 P2P 搜索比 Web 搜索要优越的地方。

3.3　P2P 搜索技术综述

P2P 搜索作为 P2P 应用的核心技术，其目标是在 P2P 这种分布式的动态环境中以最快的速度找到最多的满足用户要求的系统结点资源。要搜索到这些资源，就需要研究结点的组织方式、资源的组织方式和信息组织方式等多方面的关键技术；所以，在细致讲述 P2P 搜索技术之前，有必要了解一些 P2P 搜索领域的相关知识，包括 P2P 搜索要研究的内容、P2P 搜索研究的现状及当前主流的 P2P 搜索方法等。本节重点对 P2P 搜索技术方面的知识进行简要的综述性介绍。

3.3.1　P2P 搜索研究的内容

P2P 搜索本身是一个比较广泛的概念，从理论研究到应用实践均涉及了多方面的内容。从 P2P 搜索技术的一路发展来看，P2P 搜索技术研究的内容总体来讲主要包括以下几个方面。图 3.5 展示了 P2P 搜索技术研究的内容体系。

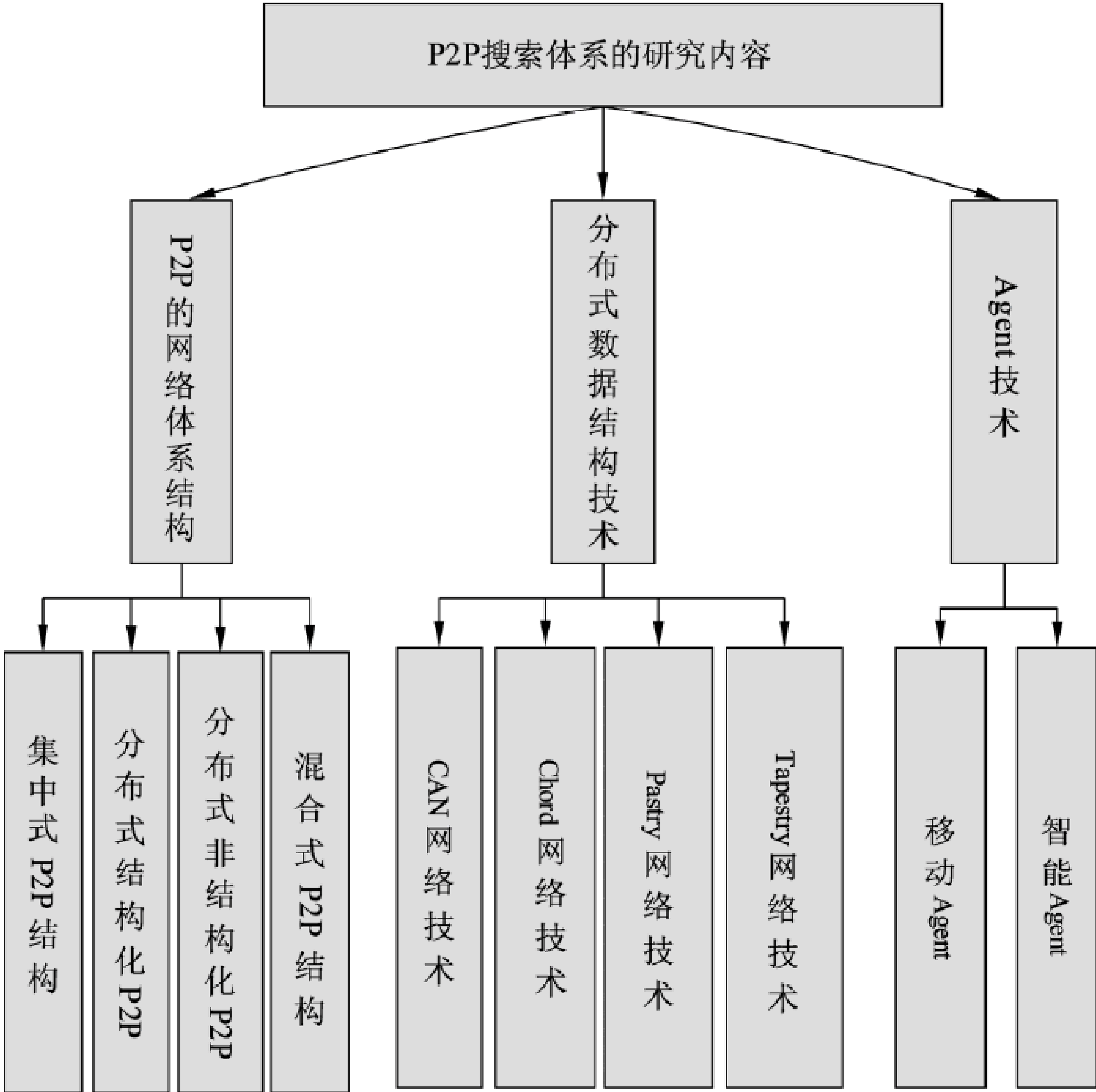


图 3.5　P2P 搜索技术研究的内容体系

1. P2P 网络体系结构

搜索算法设计和体系结构设计相辅相成，具有统一性，因而对 P2P 网络体系结构的研究是 P2P 搜索算法研究的前提和基础。

体系结构，指的是系统由哪些部分组成及这些部分之间的组织方式。针对 P2P 体系结构而言，P2P 系统由一系列地位对等的结点组成。结点的组织方式，主要体现在元信息的管理方式和获取方式上，而 P2P 搜索往往直接依赖于可使用的信息。对第一代无结构 P2P 搜索技术的研究主要在网络拓扑结构、消息路由策略、数据复制策略和缓存索引策略等几个方面。基于 DHT 的第二代结构化 P2P 系统，研究的内容主要在于如何选取拓扑结构和路由算法，如何增强对复杂性的支持，如何解决物理邻近性等问题。

注意：P2P 网络结构及拓扑的相关知识在本书第 2 章已有详细的讲解，P2P 的网络体系结构是研究 P2P 搜索的基础和关键。

2. 分布式数据结构

利用分布式哈希表算法来管理分布式资源和路由信息，比目录管理模型有很大改进。和中心结点服务器不同，DHT 网络中的各结点并不需要维护整个网络的信息，而是只在结点中存储其邻近的后继结点信息，这就大幅减少了带宽的占用和资源的消耗。

基于 DHT 构造的结构化覆盖网络不需要中心结点服务器，每个客户端负责一个小范围的路由，并负责存储一小部分数据，从而实现整个 DHT 网络的寻址和存储。因而要实现一个良好的基于 DHT 的 P2P 搜索算法，就要细致地探索研究 DHT 这类可用于组织可扩展的分布式系统的数据结构、机制或算法。

3. 智能 Agent 和移动 Agent

移动 Agent 是一个能在异构网络中自主地从一台主机迁移到另一台主机，并可与其他 Agent 或资源进行交互的程序。Agent 非常适合在网络环境中来帮助用户完成信息检索的任务。在 P2P 软件中嵌入移动 Agent，尤其是智能 Agent，在个人助手、智能搜索、实现灵活性等方面将具有很大优势，也是 P2P 搜索要研究的内容之一。

注意：Agent，字面解释为“代理”的意思，主要应用在人工智能科学方面。人工智能学科的创始人之一 M.Minsky 曾经在 1994 年就指出，“Agent 是一些具有特别技能的个体”。针对计算机系统，Agent 是指“当使用者向机器说明完成某些任务，而无须了解机器自身是如何工作，即将其处理为黑箱时，就称其为 Agent”。

3.3 2　P2P 搜索的国内外研究现状

作为 P2P 系统的核心技术，以解决跨平台、多格式、深度的信息检索为目标的 P2P 搜索技术，从其诞生那天起就是一个热门的话题，是一个业界研究的热点之一。

P2P 的研究首先是对 P2P 网络拓扑模型的研究，因为拓扑是算法执行的平台，是算法设计的基础，两者是密切相关的。到目前为止，拓扑结构主要有以下 4 种形式：中心化拓

扑、全分布式非结构化拓扑、全分布式结构化拓扑（也称作 DHT 网络）和半分布式拓扑。

🔔注意：关于 P2P 拓扑模型的相关知识，请参考本书第 2 章所讲的内容。

在国外，开展 P2P 研究的学术团体主要包括 P2P 工作组（P2PwG）、全球网络论坛（Global Grid Forum，GGF）等组织。

🔔注意：P2P 工作组成立的主要目的，是希望加速 P2P 计算基础设施的建立和相应的标准化工作。P2PWG 成立之后，对 P2P 计算中的术语进行了统一，也形成相关的草案，但是在标准化工作方面工作进展缓慢。目前 P2PWG 已经和 GGF 合并，由该论坛管理 P2P 计算相关的工作。GGF 负责网格计算和 P2P 计算等相关的标准化工作。

在企业的研究方面，Microsoft 公司成立了 Pastry 项目组，主要负责 P2P 计算技术的研究和开发工作。目前 Microsoft 公司已经发布了基于 Pastry 的软件包多个 P2P 应用。Intel 也发布了基于.Net 基础架构之上的 P2P 加速工具包和 P2P 安全 API 软件包，从而使得微软.NET 开发人员能够迅速地建立 P2P 安全 Web 应用程序。在国内，迅雷公司揉合了 P2P 和 CS 架构，成为“高速下载”的代名词，并发展成为全球最大的下载引擎。此外，如 POCO、Kugo 等这类 P2P 软件也推出了自己的搜索产品，整个 P2P 行业正在全球蓬勃发展。

与此同时，各类拓扑下的搜索算法也如雨后春笋般出现。在结构化方面，例如，微软研究院提出的 Pastry，MIT 提出的 Chord 算法和 GRID 算法等，AT&TACIRI 中心的 CAN 算法，都是这些优秀算法的代表。非结构方面，基于转发机制的宽度优先搜索（BFS）及各种改进的 BFS，反复深入算法，随机漫步搜索算法和各种智能的算法如基于人工免疫原理的搜索算法等，也相继出现。此外，还有基于缓存方法的本地索引技术，以及基于拓扑结构的优化的 GIA 拓扑自适应机制等，都步入了研发的行列。

P2P 搜索引擎从其出现的那天起，其发展道路就异常坎坷，并且直到现在也未能成为主流的搜索引擎，但不管怎么样，P2P 搜索引擎的研究依然代表着 P2P 发展的方向。2007 年 1 月，全球最大的搜索引擎公司 Google 注资 500 万美元给中国最大的下载服务提供商迅雷技术有限公司，标志着 P2P 搜索的商业化的成熟契机已经到来。目前，用户数量已经突破 1.2 亿，P2P 搜索在两大巨头的联手推动下将会成为主流的综合性搜索引擎。

3.3.3　现有主要的 P2P 搜索方法

尽管 P2P 行业发展迅速，出现各种各样的搜索算法，但各种算法均有优势和不足，还没有哪一种算法可以成熟到完全取代其他，成为最佳的算法。就目前 P2P 搜索技术的发展来看，在搜索方面上主要有基于结构化 P2P 网络的搜索和基于非结构化 P2P 网络的搜索。它们依据自身的特点和应用，又有很多种不同的算法变种。

在非结构化的 P2P 网络中，网络模型中的结点不需要了解整个网络的拓扑结构，也不需要维护复杂的网络结构图，因此在搜索算法上主要就是基于广播式的泛洪搜索。在此基础上根据应用的需要和算法的不断改进，又出现了诸如迭代加深搜索、随机漫步搜索、有向的广度优先搜索、本地索引搜索、路由索引搜索、基于移动代理搜索等多种搜索技术。

在松散结构的 P2P 网络中，比如 Freenet 网络中的搜索、幂律分布网络中的搜索等，

也都属于非结构化 P2P 网络的搜索技术。

在一个严格结构化的 P2P 网络系统中，结点和数据存放位置的相邻关系是严格定义的。这种定义是基于分布式哈希表——DHT 技术来进行的。在 DHT 技术中存在两个非常重要的概念，即目标位置和路径位置。目标位置涉及的主要内容是数据项要被存储在紧邻使用者的地方，一个给定系统的结点将数据项储存在同一个系统里。路径位置则是指查询结点和对该查询负责的结点之间的路径，需要特别指出的是这两个结点必须在同一个系统里。根据这两个重要概念，研究人员设计出了多种基于 DHT 的 P2P 网络模型，并形成了依赖于此结构的多种搜索技术，如 CAN 网络、Pastry 网络、及 Chord 网络中的信息查询技术等，都是基于 DHT 技术在结构化 P2P 网络中的查询方法。

说明：在 P2P 网络中，除了以上几种主要搜索方法外，还有其他很多种根据实际需求演化来的搜索算法及改进算法，本章余下的内容就会对这些算法进行详细的讲解。

3.4　P2P 搜索技术评价标准

有句话说，没有规矩，不成方圆，对 P2P 搜索技术的评价也需要一套规范。上文已经讲过，当前 P2P 的搜索技术有很多种，每一种都有其应用的价值，每一种都有其存在的意义。但是当前任何一种 P2P 搜索技术也无法适用所有的 P2P 网络，不仅仅是现在，在可预见的将来，也都会是这样。理解这些评价标准对设计优良的 P2P 搜索算法有重要的意义。

3.4.1　P2P 搜索的评价指标体系

从目前 P2P 搜索技术的应用和发展来看，不同的 P2P 搜索算法的设计目标不尽相同，因此衡量其优劣的方式和角度也不唯一。但从总体上来看，大多数搜索算法的评价还是集中在几个方面，比如要从用户的角度看它的适用性如何、从网络的角度来看它的可用性怎么样等。如图 3.6 所示的就是关于 P2P 搜索的评价指标体系图。

3.4.2　从用户的角度评价 P2P 搜索

好的搜索技术可以使用户能够有效地定位需要的内容，用户评价一个搜索技术的好坏通常从以下几个方面来评价。

1．对搜索结果的数量评估

用户执行一个 P2P 搜索，最直接、简单的评价方式就是对搜索结果数量的评估，执行同一个搜索，使用搜索算法 A 可以比搜索算法 B 得到更多的可用搜索结果，那么 A 显然是优于 B 的。这也就是说，在 P2P 搜索上，用户往往关心的是，一次查询能搜索到多少可用的 Peer，能找到多少适用的结果。

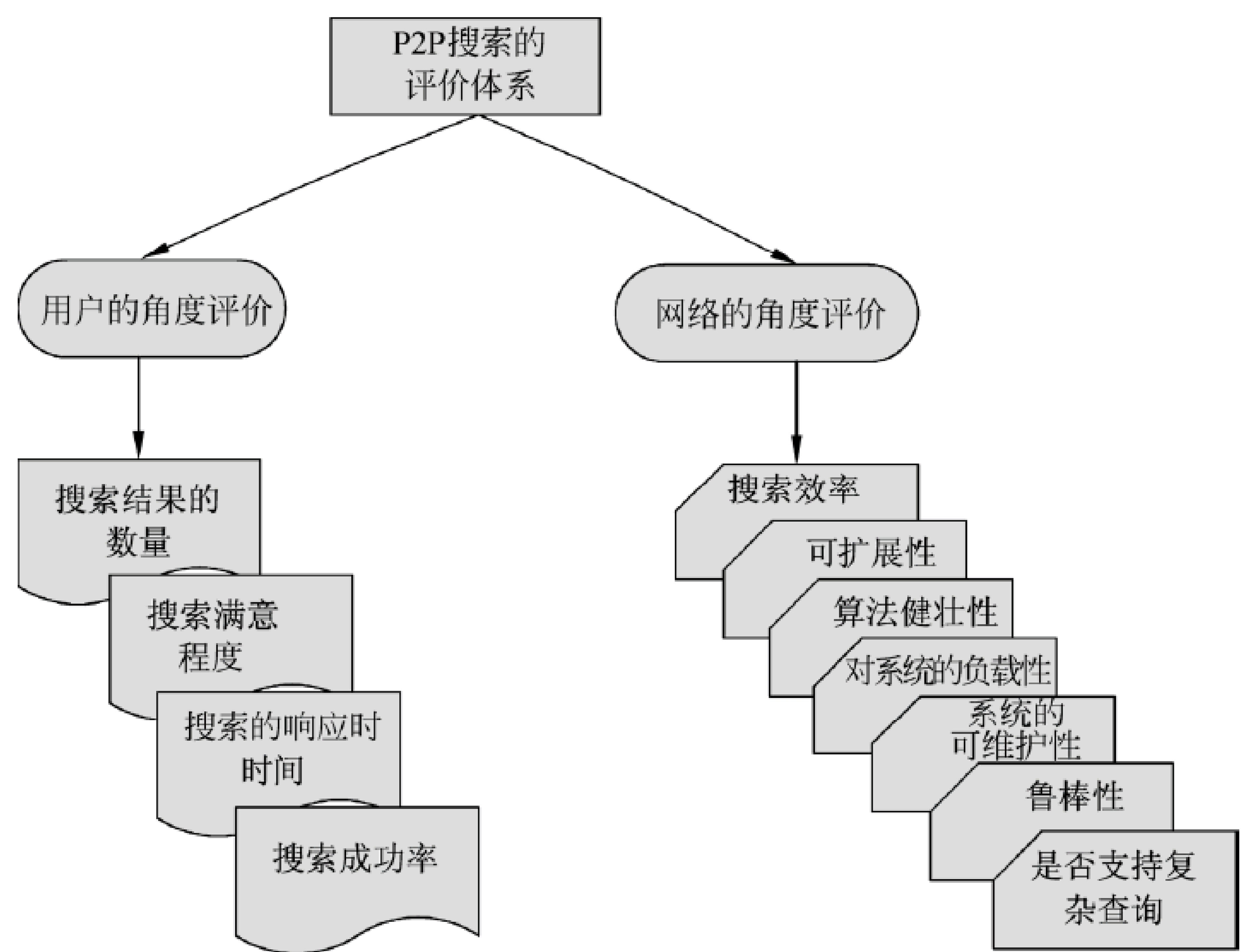


图 3.6　P2P 搜索技术的评价体系

2．对搜索的满意程度

有时候一次查询能够找到许多结果，许多系统（如 Napster，Gnutella）并不显示所有的结果，而只是显示其中的 Z 个结果，Z 是用户定义的一个值。如果查询能够找到 Z 个或者更多结果，我们就说查询是令人满意的，否则就是令人不满意的。这个满意的程度有时候直接决定用户的查询是否成功。如果在一次搜索显示出的 Z 个结果中，没有用户想要的内容，显然这次搜索是不满意的，本次查询操作也就失败了。所以，一个好的搜索算法应该尽可能地满足用户的查询需要。

3．对搜索的响应时间

这里的响应时间指的是用户从发起查询开始到收到 Z 个结果需要等待的时间，这个时间的大部分是消耗在查询过程中的，这与 P2P 搜索中平均查询“跳”数有直接关系。平均查询跳数（Average Query Hops）也称为平均搜索跳数，是指由源结点发出对某个资源的查询请求后，查询请求消息平均需要经历多少跳（hop）的转发后源结点才能得到“命中”或“搜索失败”的响应。它影响到用户的响应时间，属于用户性能的一个指标。

注意： 对于结构化 P2P 搜索算法来说，平均查询跳数是 O(logn)；对于无结构 P2P 搜索算法来说，如果是随机广播路由（如 Gnutella），那么其平均查询跳数也是 O(logn)，如果是随机单向路由（如 Freenet），那么其平均查询跳数存在不确定性，但要高于 O(logn)。

4．搜索的成功率

搜索的成功率（Query Success Rate）是指当被查询的资源在网络中已经存在的情况下，

每次查询获得命中响应的概率。如果资源存在但第一次查询未命中，源结点就需要再次进行查询，或者误认为网络中不存在该资源从而放弃查询。可见查询成功率影响整个搜索算法的效果，如果在实际应用中查询成功率太低，会直接影响用户的使用。

相比而言，在结构化 P2P 搜索算法中，查询的成功率相对较高，只要网络中确实存在被查询的资源，那么通过存储映射总能命中，查询成功率为 100%。而非结构化的 P2P 搜索算法通常情况下为了防止查询请求消息包在网络中无限期地传播，设置了生存时间（TTL）减值的机制。如果经过 TTL 个查询“跳步”后还没有搜索到所请求的资源，查询请求消息包会被丢弃，所以查询成功率达不到 100%。

注意：关于本节中提到的结构化 P2P 搜索算法和非结构化的 P2P 搜索算法都是下文要重点讲述的知识，可参考 3.5 节以后的相关内容。

3.4.3　从网络的角度评价 P2P 搜索

从网络的角度来评价搜索技术，主要是从搜索算法本身的特性及效能来进行评估，一般包括搜索的效率、搜索技术的可扩展性、健壮性等。

1．搜索效率

查询请求信息在网络中扩散，要经过许多结点。每个结点都要花费处理资源（如 CPU 时间、存储空间等）对请求进行处理，这种处理主要是验证是否有满足查询要求的内容，是否要转发查询请求。另外，当结点接受或转发查询请求以及发送响应消息时，也消耗了带宽资源。往往用资源的消耗来衡量搜索的效率，资源主要包括带宽、CPU 的处理时间、存储空间等。高效率的搜索技术要尽量以最少的资源消耗获得最满意的搜索效果。

2．可扩展性

P2P 网络的规模一般都很大，而且随着加入的用户越来越多，规模快速增长。P2P 搜索方法必须能够适应网络对可扩展性的要求，在网络规模急剧膨胀的时候也要能够及时有效地搜索到用户需要的资源。

3．健壮性

P2P 网络从来就不是静止不动的，P2P 网络中的结点会自动地、频繁地加入或退出。根据对 Gnutella 的统计，40%的结点在线的时间少于 4 个小时，只有 25%的结点在线时间超过 24 小时。对于这种动态的 P2P 网络，搜索方法应该具有良好的健壮性，在结点出现故障或者退出网络时，将影响降到最低，尽可能不影响对其他结点的搜索，保证搜索到足够的资源。

4．系统的负载性

搜索算法的系统负载性多出现在非结构化的 P2P 搜索算法中，与其平均查询包数（Average Query Packets）有着直接的关系。在一次完整的搜索过程，无论结果是命中或不命中，P2P 搜索算法都会向 P2P 网络中发送大量的查询包。这些在 P2P 网络中传播的查询

请求消息包必然会影响到系统的通信负载，如果是在基于泛洪机制或在泛洪基础上改造而来的算法，其平均查询包数通常较大，系统的通信负载也较重。

5．系统可维护性

系统可维护性指维护一个 P2P 网络系统的拓扑结构的难易程度，属于系统性能的一个指标。结构化搜索算法因为网络的拓扑结构限制较大，维护代价很高，所以系统可维护性较差；而无结构搜索算法因为网络并不存在强硬的结构，几乎不需要什么维护代价，所以系统可维护性很好。

6．鲁棒性

在一个高度动态的 P2P 网络系统中，结点会随机地加入或退出网络，而且这种行为发生的频率可能很高，导致了网络拓扑时常在变化。在这种情况下，搜索算法的性能与静态网络系统环境时相比，会有多大的影响，称为这种搜索算法的鲁棒性。影响越小，说明算法的鲁棒性越高。一般来说，无结构搜索算法较结构化搜索算法的鲁棒性比较高。

7．是否支持复杂查询

结构化 P2P 搜索算法建立在分布式哈希表的基础之上，通过信息到结点的映射来进行搜索，所以只能处理精确的关键字，无法支持复杂化的查询请求。无结构 P2P 搜索算法原理与其不同，可以支持复杂查询。

以上评价指标并不独立存在，而是互相制约的。比如通常情况下，平均查询跳数较小的搜索算法（意味着响应用户的查询请求较快，用户性能比较好），其平均查询包数一般也较多（意味着网络通信负载较大，系统性能比较差）。所以必须在搜索算法的设计过程时做出一个良好的折衷，为了适应不同的应用场合，也有可能需要偏重于某个或某些指标。

在 P2P 不断发展的过程中，还会有更多的、新的 P2P 搜索技术产生，也有很多现有的技术不断被淘汰。了解了这一套评价体系，就能设计出更好、更有应用价值搜索算法。

3.5　集中式 P2P 网络搜索技术

P2P 的搜索技术与其结构是有密切联系的，有什么样的 P2P 网络拓扑基本上也就决定了在此网络拓扑下的搜索策略。集中式 P2P 网络的结构在第 2 章中已经做了详细的讲解，基于这个结构的搜索技术都叫做集中式的 P2P 搜索技术。

3.5.1　集中式 P2P 的目录索引机制

集中式 P2P 的结构如图 3.7 所示，在这一结构中，主要采用目录索引的机制来发现 P2P 网络中的结点。

集中目录式 P2P 结构是最早出现的 P2P 应用模式，因为仍然具有中心化的特点也被称为非纯粹的 P2P 结构。如图 3.7 所示，在集中式的 P2P 网络结构中，对等体的查询发送到一个单一的索引服务器。索引服务器根据本地保存的客户端资源索引，对查询做出反应。

当各个对等体的资源出现变化时，比如资源的增加、删除等，索引服务器将收到这些变化的新消息，并据此修改本地缓存，但查询信息不在客户端的对等结点间传递。

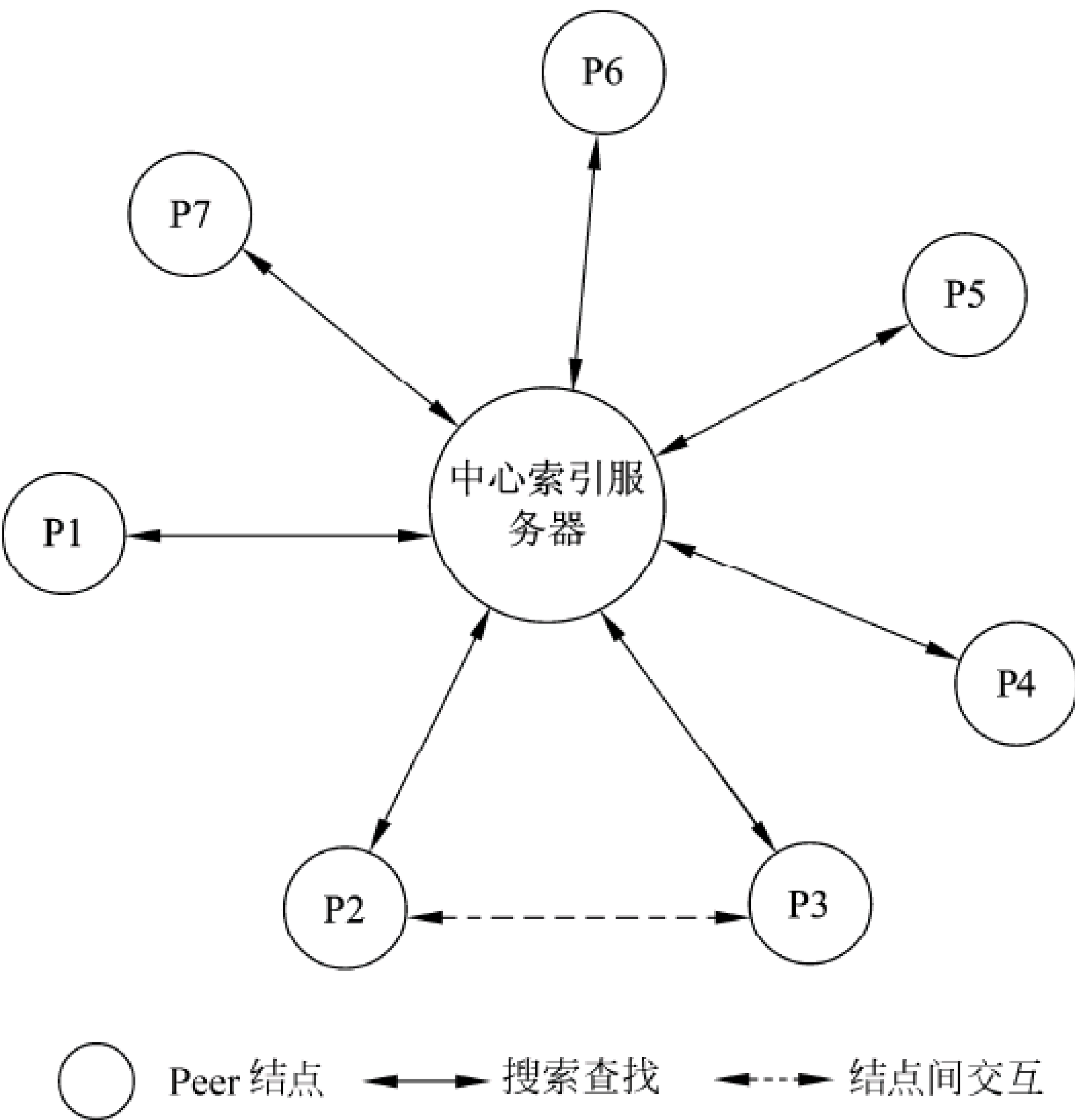


图 3.7　集中式 P2P 搜索的目录索引机制

在索引目录发现机制模型中，一台或多台有特殊用途的服务器为对等点提供目录服务。对等点向目录服务注册关于自身的信息（其名称、地址、资源和元数据）。如图 3.7 中，从 P1～P7 的 7 个 Peer 结点分别独立的与索引服务器通信，以注册自身的信息注册完成后，中心索引目录服务器上便存储有各个 Peer 结点的信息。

在查询发生的时候，Peer 结点根据目录服务器中信息进行查询，通过目录服务器来间接地定位其他对等点，图 3.7 中 P3 与 P2 的通信就是通过索引服务的媒介作用来完成的。当对等体的资源出现变化时，比如资源的增加、删除等，索引服务器将收到更新消息，并据此修改缓存；但是此类系统中的数据存储仍是分布的，信息资源的传输也是点对点的，不需要服务器的介入。

3.5.2　集中式 P2P 的搜索策略

集中式搜索是基于一台或多台中央索引服务器来进行的。中央索引服务器负责协调或者调度单独注册结点上的资源。一般地，中央服务器维持着 P2P 网络中结点的源的中央目录，并协调结点间的交互。

在集中式 P2P 中，当执行一次搜索的时候，所有的结点都向中央索引服务器发送请求信息进行查找，而不向其他结点提出请求。索引服务器就像一个巨大的“电话簿”，存储着各个结点的联系信息，这些信息包括结点信息、资源信息等。当查询到达索引服务器后，牵引服务器便开始在自身庞大的信息库里搜索与查询资源相关的信息，一旦找到这些信息，牵引服务器便把注册这些资源信息的结点“联系方式”返回给查询结点，这样两个对等结点之间就可以相互通信了。

对等结点通过中央服务器来完成资源的定位通信信息，一旦对等结点之间建立起了连接，索引服务器就不再参与对等结点之间的实际通信，它只负责把请求结点和目标结点联系起来，而真正的文件传送是在源请求结点和目标结点之间直接进行的。当结点的资源发生变化时，比如资源的增加、修改、删除等，索引服务器将收到更新消息。并据此修改服务器缓存的资源索引信息。如图 3.8 描述的是集中式 P2P 的索引目录的发现机制。

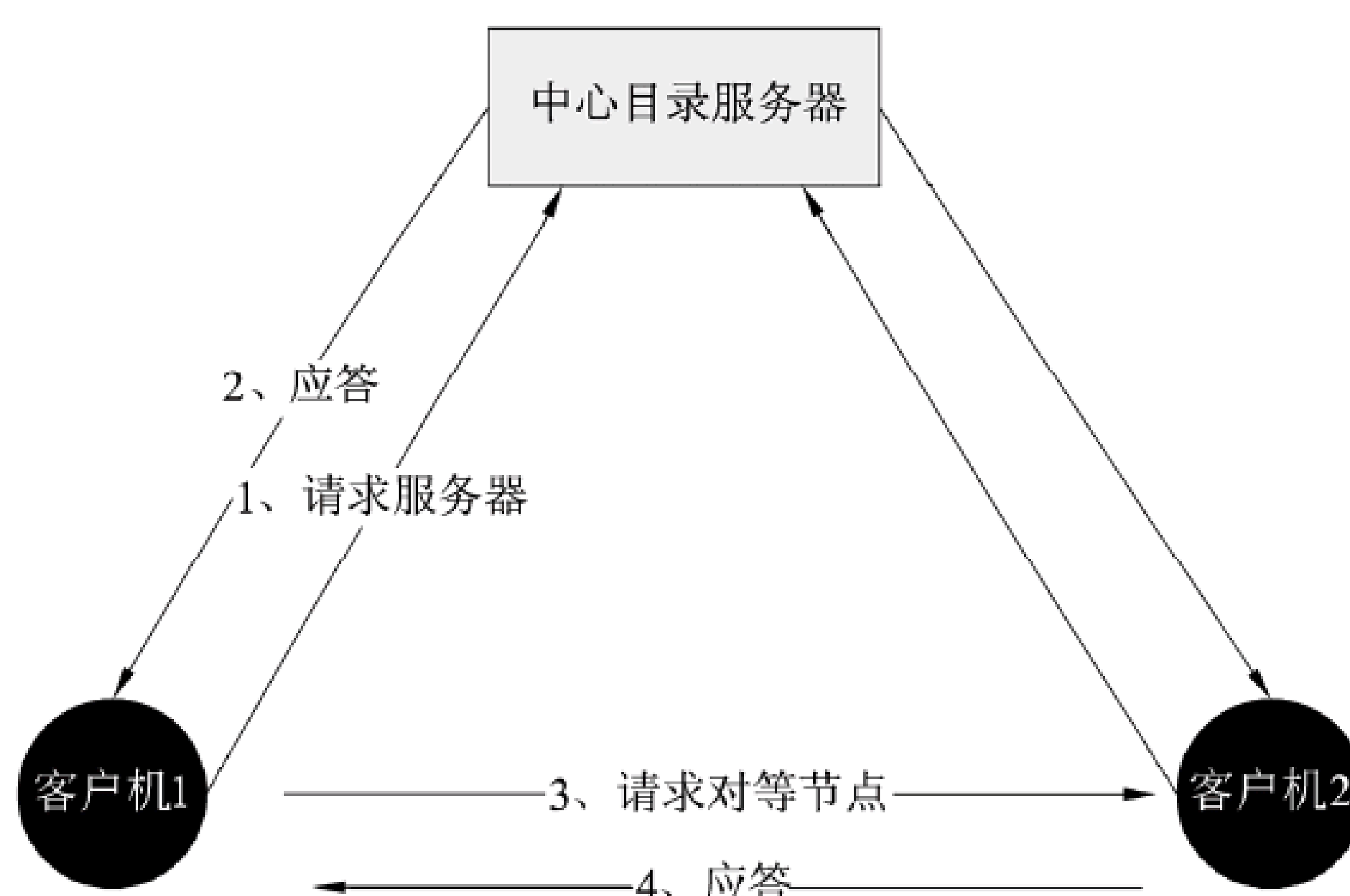


图 3.8　索引目录的发现机制

图 3.8 展示了一个使用目录服务器向对等点提供位置和命名服务的 P2P 体系结构。客户端客户机 1 向目录服务器来发送定位资源信息的请求，目录服务器告诉客户机 1 所需要的资源位于另一客户机 2 上，然后客户机 1 就直接向目标客户端客户机 2 发送请求建立连接来获取资源，这样真正的数据传输是不需要目录服务器干预的。可以说这种方式就是借用传统的 C/S 模型的发现机制来实现 P2P 网络模型下的资源定位。不过这种模型受到了“非 P2P”的指责，因为其背离了 P2P 的平等理念。用于共享 MP3 音乐文件的 Napster 是其中最典型的代表，也正是它引发了网络的 P2P 技术革命。

3.6　结构化 P2P 网络的搜索方法

结构化 P2P 网络中，每个结点都有固定的地址，整个网络具有相对稳定而规则的拓扑结构。依赖拓扑结构可以给网络的每个结点指定一个逻辑地址，并把地址和结点的位置对应起来。在实际的系统中，P2P 网络的逻辑地址通常是由 Hash 函数得到的，每个结点都保存一张 DHT（Distributed Hash Table）表进行路由，所以结构化 P2P 网络通常也叫做 DHT 网络。关于 DHT 网络及它们的拓扑结构及路由和查询方法在第 2 章中已经做了介绍，这里不再赘述。下面重点讲解的是它们对结点的发现算法，也就是它们对资源的搜索方法。

注意： 关于 DHT 的相关知识，请查阅第 2 章的有关内容。

3.6.1　基于 DHT 的资源定位机制

结构化的 P2P 系统一般都是基于 DHT 技术的，运用 DHT 技术建立具有一定结构的逻

辑拓扑，使结点与文件之间建立一定的关系，每个结点按一定规则保存系统中部分其他结点的信息，为文件的搜索提供一定的信息。

在结构化的 P2P 系统中，利用 DHT 机制来对 P2P 系统中的文件进行定位时，一个文件与一个 key 值相对应。key 值一般通过对文件进行哈希得到，系统中的每个结点负责保存一定范围的 keys。不管内部的搜索算法如何，应用接口均由 put(key,value)和 get(key)这两个函数组成，其中 put(key,value)的功能是进行结点的信息发布，get(key)的功能是进行信息查询。这样，通过这种机制可以对 P2P 网络中的文件进行定位，在需要搜索某一文件时，只需执行 get(key)功能，便可以进行一次搜索查询。这就是基于 DHT 的资源定位机制。

3.6.2　基于 DHT 的搜索实现

上面讲述了 DHT 对资源的定位机制，而要通过 DHT 实现对资源的搜索，还需完成以下几个关键点。

1．DHT的建立与键值对的产生

散列表的建立及散列值的产生都需要使用一个基本的散列函数，比如 SHA-l 函数，结点的标识符采用结点名字（比如 IP 地址）的散列值；对象的标识符采用对象名的散列值；每个结点存储一张散列表，存储结点标识符与结点物理地址的映射。

2．内容的查找

内容查找通过<key,value>对来查找，key 在这里对应着对象标识符，value 在这里对应文件的名称、所在的结点的地址等信息。

3．定位关键字所在的结点

将各个结点所具有的<key,value>对保存在与对象标识符相近的结点标识符的机器中，使搜索关键字代表的对象与结点地址关联上。

4．<key,value>键值对的流动

当有新结点或新的<key,value>键值对出现时，将对应的<key,value>键值对转移到对应的结点上；当旧结点离开时，将其储存的<key,value>键值对转移到相邻的结点上。

目前典型的 DHT 网络包括 chord、Pastry、CAN 和 Pastry 等，它们的主要区别在于采用了不同的 DHT 路由算法，这也决定了各类网络在逻辑拓扑上的不同。这些系统建立在确定性拓扑结构的基础上，从而表现出对网络中路由的指导性和网络中结点与数据管理的较强控制力。下面就较重要的几个系统模型进行介绍。

3.6.3　Chord 网络搜索技术

Chord 采用一维的环形拓扑结构，关键字和结点在同一个标识符空间表示，都用 m 位的标识符表示。Chord 运用一致性散列函数（Consistent Hashing）将关键值分配给结点。一致性散列函数为每个结点和关键值赋予一个 m 位的标识符，结点的标识符是通过对结点

的 IP 地址进行散列计算得到的；关键字值的标识符是通过对结点中存储的关键值（如某个可被查找文件的关键值，即在查找此文件时可用来表示此文件特征的关键字信息）进行散列计算得到的。在计算标识符的时候，其标识符的长度应该取的足够长，这样任意两个不同的结点，它们的关键值在径散列计算后得到相等标识符的可能性就很小。

当关键值和结点经过散列函数计算后，关键值通过如下方法存储在相应的结点中。关键值 k 被存储在一个结点值等于 k 的结点上，如果不存在这样的结点则存储在结点值大于 k 的第一个结点上。存储关键值 k 的结点称为关键值 k 的后继结点。结点按照结点值从 1～2^m 的形式排成一个环，那么 k 的后继结点就是 k 在顺时针方向上的第一个结点，该环称为 Chord 环。如图 3.9 所示的就是一个基于 DHT 技术的 Chord 环。

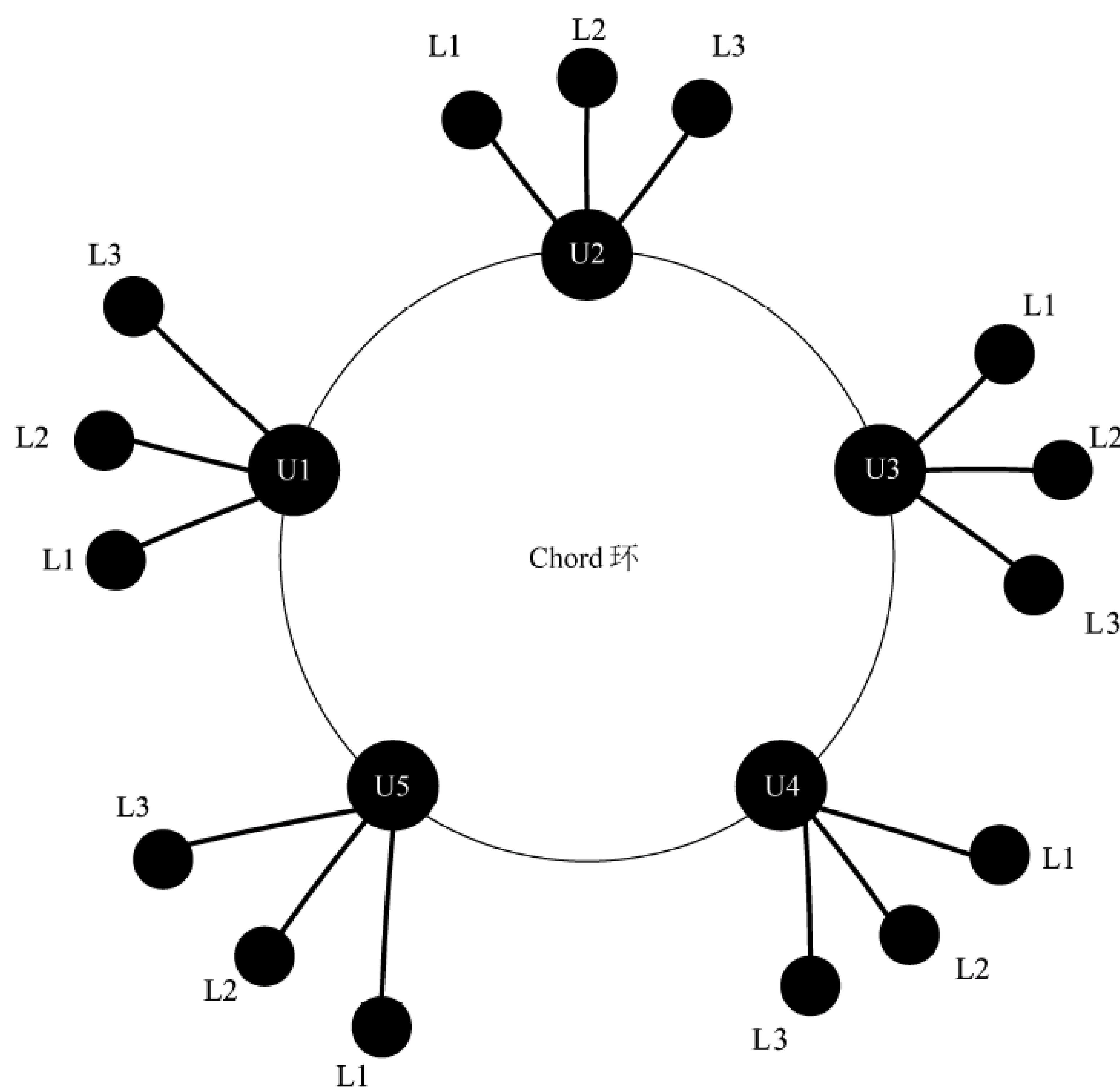


图 3.9　基于 DHT 技术的 Chord 环

在 Chord 的路由模型中，它的路由过程中每个结点只需要知道在 Chord 环中它的后继结点是谁就行。查询过程是给定的关键字沿 Chord 环通过后继结点的指针进行传递，直到遇到一个结点的标识符数值超过这个关键字标识符。

这种查询方法效率并不是很高，如果网络中有 n 个结点，那么就需要跨越 n 个结点来找到关键字和结点的映射。为了改进查询的速度，Chord 增加了额外的路由表 Fingertable 来加快查询进度。FingerTable 如表 3.1 所示。

在 Chord 网络中进行定位的算法具体过程如下。

（1）定位过程可以是在其他函数调用时发生，或接收到定位消息（可以定位前继或者后继结点）时发生。首先检查待定位的关键值是否等于定位过程发生结点的结点值，如果

相等，则向原函数或者信息发出者返回本结点值（如果定位前继）或者返回后继（如果定位后继）；否则转步骤（2）。

表 3.1　Chord路由模型的FingerTable表

查 询 表 项	定　义
Finger[k].start	（n+2k-1）　mod 2m，　1≤k≤m
.interval	（finger[k].start，finger[k+1].start）
.node	first.node ≥ n.finger[k].start
next	在环上离本地结点最近的后一个结点，也就是 finger[i].node
previous	在环上离本地结点最近的前一个点

（2）检查待定位关键值是否在过程发生结点的后继之上，即关键值是否大于过程发生结点却小于过程发生的结点的后继。如果关键值并不在后继之上就转向步骤（3）；否则，向调用过程返回（如果是本地调用），或者远程结点返回本结点值（如果定位前继）或者返回后继（如果定位后继）。

（3）检查 finger 表中的 node 是否有处于待定位关键值和本身结点值之间的结点值。

注意：不能取 Finger 中 node 等于关键值的结点值。

如果 finger.node 中存在这种结点值就向这些值中最接近关键值的结点发定位消息；否则从 finger.node 中找到最接近被定位关键值的结点值，并向这个结点发送消息。接收消息的结点再重新从步骤（1）开始这个流程。

以上 3 步，就是在 Chord 网络中进行结点定位的算法实现过程，在这一过程中需要重点注意查询 finger 表的方法。

3.6.4　Pastry 网络搜索技术

Pastry 在 2001 年由微软研究院和莱斯（Rice）大学共同提出，同 chord 一样，它也是在 DHT 网络下的一个变种。

在 Pastry 协议中没有规定具体应该采用的散列算法，但是定义了散列值为一个一维空间，在实际应用中，这个一维空间就是 128b 的整数空间。根据 Pastry 协议中，每个结点都拥有一个 128b 的标识。为了保证标识的唯一性，一般由结点的网络标识，如 IP 地址和端口等，经过散列得到。

Pastry 中的每个结点都要维护一个数据结构，这个结构性的数据拥有一个路由表，一个邻居结点集合和一个叶子结点集合，它们一起构成了结点的状态表。

在 Pastry 中发起一个查询的过程如下。

（1）路由查询消息携带被查询键值，执行查询过程，路由消息发送给结点，执行步骤（2）。

（2）当收到路由消息时，结点首先检查该键值是否落在叶子结点集合的范围内。如果是，则执行步骤（3），否则执行步骤（4），如果不存在这样的结点，则执行步骤（5）。

（3）结点直接把消息转发给叶子结点集合中结点的标识和消息的键值最接近的结点，跳转到步骤（6）。

（4）结点从路由表中根据最长前缀优先的原则选择一个结点作为路由目标，转发路由消息，跳转到步骤（6）。

（5）不存在这样的结点时间，当前结点将会从其维护的所有邻居结点集合中，包括路由表叶子结点集合及邻近结点集合中的结点。选择一个距离该键值最接近的结点作为转发目标，将查询消息转发出去，执行步骤（6）。

（6）另一个结点在接到路由消息后，从步骤（2）开始，执行同样的操作。

3.6.5　CAN 与 Tapestry

内容寻址网络（Content Addressable Network，CAN）在 2001 年由加州大学伯克利分校提出，CAN 是在现有网络上抽象出的一层叠加网络。

在 CAN 中将网络的结点映射到一个 d 维的笛卡儿空间中，并尽可能为每个结点均匀分配一块区域。CAN 采用的哈希函数通过对关键值进行 Hash 运算得到笛卡儿空间中的一个点，并将关键值对存储在该点所在区域的结点中。CAN 的路由算法相当简单直接，知道目标结点后就将请求传给当前结点所有的邻居结点中最接近目标的结点。

Tapestry 是由加州柏克莱大学所提出的点对点搜寻架构，它的路由过程与 Pastry 非常类似，同样采用的是多级匹配的算法。

在 Tapestry 中，系统中的每个结点，将自身的结点信息和存储的文档信息通过哈希变换得到各自 160 个比特的唯一标识符，每个结点通过比较收到的文档标识符和本地保存的邻居结点的标识符，并选择结点标识符和文档标识符有着最长前缀匹配的结点作为路由路径中的下一跳结点。如图 3.10 所示为结点 5230 到结点 42AD 经过的路由过程。

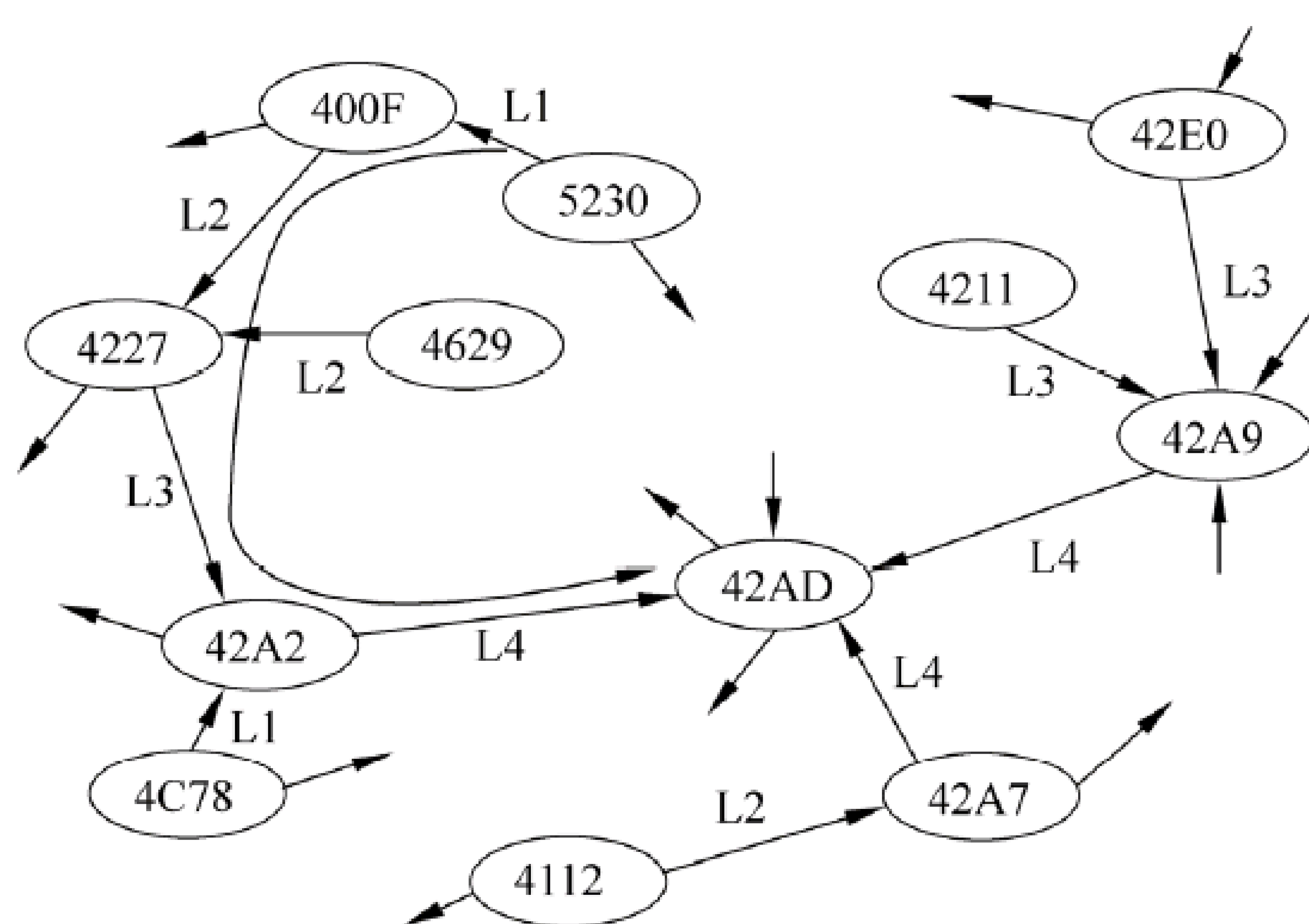


图 3.10　Tapestry 的路由示例

🔔注意：关于 CAN 和 Tapestry 的相关知识，有不明白的地方请参阅第 2 章的有关内容。

3.6.6　基于 DHT 的搜索定位技术存在的问题

正是由于不具有结构化的 P2P 系统的极不可扩展性，激发了人们对结构化 P2P 系统的

研究。基于 DHT 技术的系统基本解决了非 DHT 系统所面临的问题，如系统具有可扩展性、一定程度的负载平衡等。但 DHT 算法也存在许多问题，理解了这些问题的关键所在，就能有效地规避这些问题所带来的隐患、下面就对这些问题进行简要的分析。

1．QoS 问题

由于结点加入系统时 ID 是随机分配的，使得两个 ID 很近的结点在实际的物理位置可能很远，在 P2P 层的一跳到 IP 层就可能是多跳，这就存在请求延迟和下载延迟。目前人们开始研究在原有算法基础上考虑结点的实际物理位置，如按物理位置对结点聚集或分类等，进一步研究还有待深入。

注意：QoS 的全称是 Quality of Service，中文名为“服务质量”。它是指网络提供更高优先服务的一种能力，包括专用带宽、抖动控制和延迟（用于实时和交互式流量情形）、丢包率的改进以及不同 WAN、LAN 和 MAN 技术下的指定网络流量等，同时确保为每种流量提供的优先权不会阻碍其他流量的进程。QoS 是网络的一种安全机制，是用来解决网络延迟和阻塞等问题的一种技术。在正常情况下，如果网络只用于特定的无时间限制的应用系统，并不需要 QoS，比如 Web 应用，或 E-mail 设置等。但是对关键应用和多媒体应用就十分必要。当网络过载或拥塞时，QoS 能确保重要业务量不受延迟或丢弃，同时保证网络的高效运行。

2．拥塞问题

对于那些比较热的文件，请求和下载最后均路由到存储该文件的结点上，这就会造成这些结点的超载甚至崩溃，进而可能造成整个网络的瘫痪。目前对此问题提出的解决方法是缓存和多点复制，实验表明这在很大程度上可以解决这一问题，但还有问题存在。

3．安全问题和随机失效性

目前的 DHT 算法基本上均未考虑恶意攻击问题，这在网络中是经常存在的。恶意攻击可能给系统返回错误数据或破坏系统数据搜索功能，严重时后者可能导致系统瘫痪。现在对 P2P 的安全性问题研究还不太多，一些分布式加密开始运用到上面，还有的提出了信任度模型，但对于随机结点失效系统的恢复功能均没有采用洪泛算法的 Gnutella 性能好。

4．多关键字查询问题

DHT 算法采用分布式哈希函数，文件是通过对文件名进行哈希得到的二进制关键值，文件的查询是按关键值进行的，因此基于 DHT 的算法只支持精确查询，无法进行模糊匹配。这对用户的查询是个极大的限制，现在这方面的研究也较多，主要考虑的是数据存储方式，如建立多关键字索引等。

3.7　非结构化 P2P 网络的搜索方法

在非结构化网络中，结点没有指定的逻辑地址，采用随机方法或者启发策略加入网络，

网络拓扑随着结点的变迁和网络通信的进行而发生演变。按照搜索策略，非结构化 P2P 网络的搜索方法可以分为两大类：盲目搜索和启发式搜索。盲目搜索通过在网络中传播查询信息并且把这些信息不断扩散给每个结点。通过这种泛洪方式来搜索想要的资源。而启发式搜索在搜索的过程中利用一些已有的信息来辅助查找过程，这样可以有效地提高搜索针对性的效率。本节重点讲解非结构化 P2P 网络中的搜索技术。

3.7.1　Flooding 搜索算法

Flooding 搜索也叫泛洪搜索，在这种搜索算法中，P2P 网络中的消息像洪水一样在网络中的各个结点间流动，所以也被称为洪水算法。从算法特性上看，Flooding 算法首先遍历自己的相邻结点，然后再层次性的一层层向下遍历。在遍历的过程中，一个结点向所有邻居结点广播查询消息，邻居结点再向自己的邻居结点广播，这个过程不断进行下去。为了限制搜索的范围，消息被设置了一个初始的 TTL 值。消息每经过一个结点，TTL 值减 1。当 TTL 值为 0 时，搜索过程终止。

根据以上的描述，在 P2P 网络中采用 Flooding 搜索有以下几个特点：

（1）结点覆盖率高：一项研究表明，在 Gnutella 网络中，采用泛洪方式，能够搜索到 95%的结点。这是因为，在泛洪中，随着“跳”数的增加，结点个数也呈指数级增加。

（2）健壮性好：一个结点出现故障或是退出对其余结点几乎没有影响，因为在泛洪中，每个结点要向所有邻居结点发送消息，查询消息能够通过结点之间所有可能的路径被传送。

（3）响应时间快：在泛洪搜索中，结点是以并行的方式向邻居结点发送查询消息的，所以响应时间很快。

3.7.2　Flooding 搜索的原理及应用

Flooding 路由算法的具体路由方法及流程描述如下。

（1）当系统中的某个结点需要搜索某文件时，结点向其所有的邻居结点进行广播查询请求，执行步骤（2）。

（2）邻居结点收到请求后对照自己拥有的文件列表，将满足要求的结果返回并继续接着执行步骤（1）。

根据以上的流程，这是一个无限广播循环执行的过程，要控制广播查询的深度，就需要根据预先设定的 TTL（Time to Live）值达到 0 来终止这种查询请求广播的进行，从而终止搜索过程。

注意：TTL 是 Gnutella 系统中为了防止请求无限制地传播下去而设置的一个固定整数值。请求每经过一个结点，其跳数加 1，而其 TTL 值则减 1，当 TTL 值为 0 时则丢弃该查询请求。

在最初的 Gnutella 协议中，使用的就是 Flooding()方法，在网络中，每个结点都不知道其他结点的资源。当它要寻找某个文件时，把这个查询信息传递给它的相邻结点，如果相邻结点含有这个资源，就返回一个 QueryHit 的信息给 Requester。如果它相邻的结点都没有命中这个被查询文件，就把这条消息转发给自己的相邻结点。这种方式像洪水在网络中

各个结点流动一样，所以叫做 Flooding 搜索。由于这种搜索策略是首先遍历自己的邻接点，然后再向下传播，所以又称为宽度优先搜索方法（BFS）。Flooding()方法的示意图如图 3.11 所示。

在图 3.11 中，搜索的结点一开始时设定 TTL=3，它每传播一次，TTL 就减 1，如果 TTL 减到 0 还没有搜索到资源，则停止。如果搜索到资源则返回目标机器的信息以用来建立连接。在搜索过程中可能出现循环，但是由于有 TTL 控制，所以这个循环不会永远进行下去，当 TTL=0 的时候自然结束。

3.7.3　Flooding 搜索存在的问题

随着联网结点的不断增多，网络规模不断扩大，通过这种泛洪方式定位对等点的方法将造成网络流量急剧增加，从而导致网络中部分低带宽结点因网络资源过载而失效，查询的结果可能不完全，查询速度较慢，并由此带来可扩展性差等问题。所以在初期的 Gnutella 网络中，存在比较严重的分区、断链现象。也就是说，一个查询访问只能在网络的很小一部分进行，因此网络的可扩展性不好。

在 P2P 网络中，利用泛洪机制来传播消息，不可避免地会在网络中产生大量冗余消息，特别是当网络规模比较大、结点之间的连通度比较高的时候。以图 3.12 所示泛洪过程为例，图中包括 A、B、C、D 4 个对等结点，假设 4 个结点互相连通。如果采用泛洪机制，A 把消息发送给 B、C、D，B 收到消息后转发给 C、D，C 收到消息后转发给 B、D，D 收到消息后转发给 B、C，这是一个基本的泛洪过程。

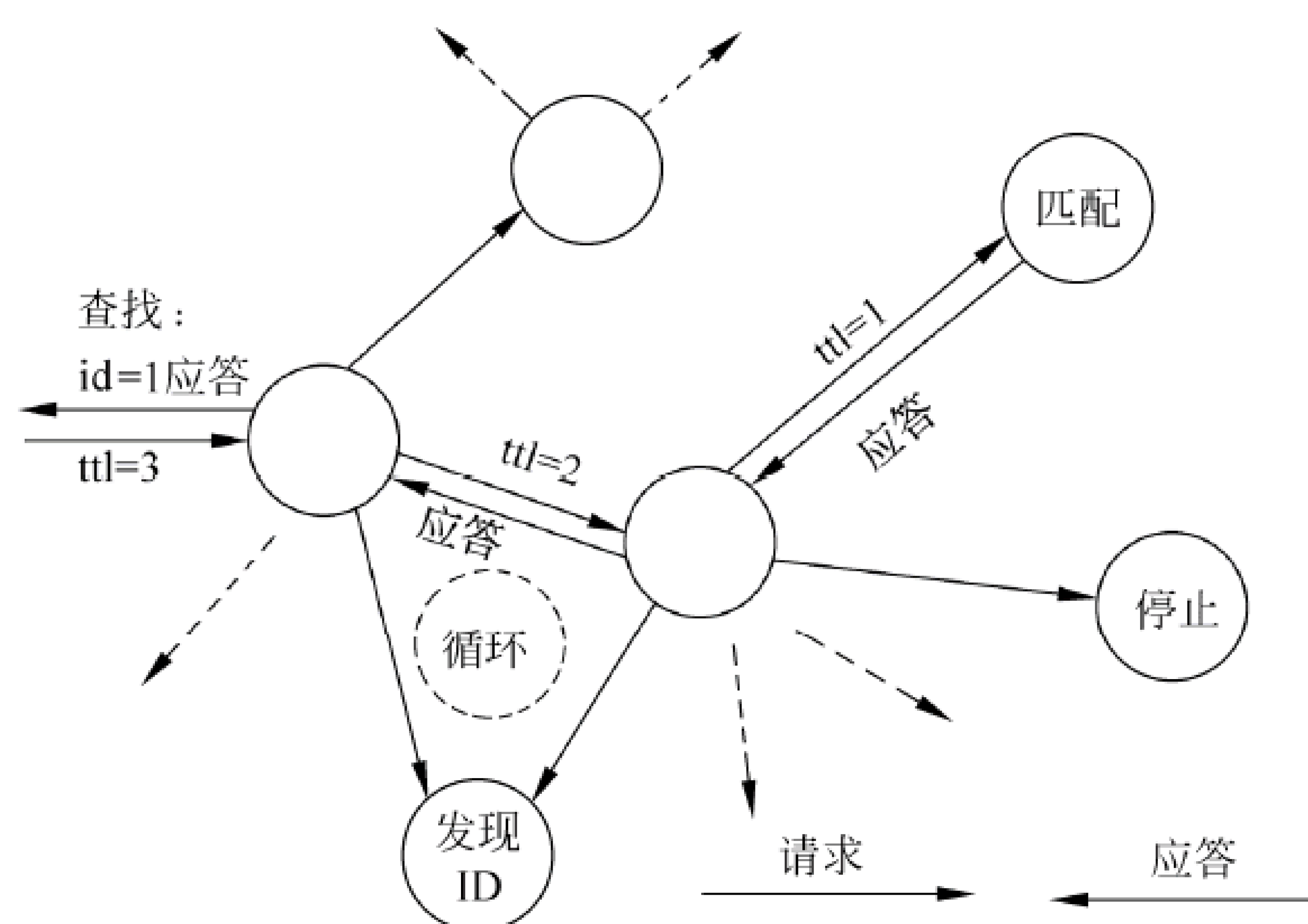


图 3.11　Flooding 方法的搜索示意图

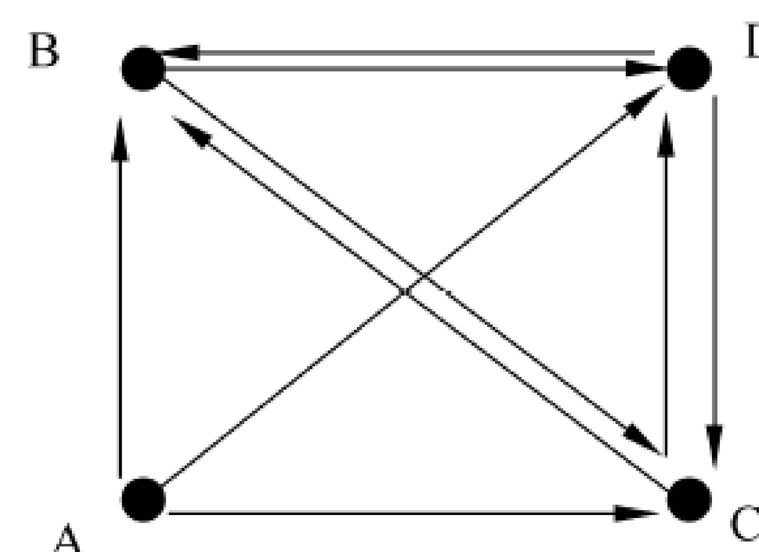


图 3.12　泛洪搜索的实例

在这个过程中，假设 B、C、D 首先接收到 A 发送的消息，然后又同时将消息进行泛洪，那么，由图 3.12 可以看出在该网络中传递的消息有 9 条，其中冗余消息就为 6 条，占消息总数的 2/3。在实际的 P2P 网络中，类似的这种冗余消息时有发生，不仅增加了结点处理负担，也占用了大量的网络带宽。

从以上分析看出，在基本的泛洪算法中，对等结点收到消息后会转发给所有邻居结点，以扩散的方式传播消息，通过设置 TTL 值控制消息在网络中的生存时间。此方法简单，可

有效地进行搜索，但在网络中产生了大量的冗余消息，限制了 Gnutella 网络在大规模系统中的应用。

所以，后来许多研究人员在泛洪的基础上作了许多改进，例如采用 Random work、Dynamic Query 等方法。

3.7.4 迭代递增搜索（iterative DeePening）

迭代递增搜索算法是对泛洪算法一种改进，它的改进点主要针对的是控制结点路由消息的扩散程度。

在迭代递增式的搜索算法中，进行多次泛洪搜索的时候，每次搜索的深度限制是递增的。当查询的结果满足要求或者已经到达最大的深度限制，过程结束。在泛洪中，随着深度的增加，结点个数指数增长。因此，如果能够在小于 D（D 为最大深度限制）的深度找到满意的结果，和直接以深度为 D 进行泛洪搜索比较起来，就不必查询很多结点，节省了大量的资源消耗。

迭代泛洪搜索原理是：首先需要决定一个策略，说明每次迭代的深度。比如说，如果想进行 3 次迭代，第 1 次的深度为 a，第 2 次的深度为 b，第 3 次的深度为 c，那么迭代策略就是 P 二{a，b，c}。因为迭代泛洪需要和深度为 D 的直接泛洪具有同样的搜索范围，所以最后一次迭代的深度值必须为 D。除了需要一个迭代策略外，还需要设置迭代之间的时间间隔 W。

以策略 P={a，b，c}为例，执行迭代泛洪搜索算法的执行流程如下。

（1）源结点首先向邻居结点发送 TTL=a 的查询请求，开始进行深度为 a 的泛洪搜索，执行步骤（2）。

（2）在深度为 a 的结点收到查询请求并处理完毕后，不是将查询请求丢掉，而是暂时存储起来。查询请求在离源结点 a 跳远的结点那里被“冰冻”起来，深度为 a 的结点成为搜索的前线结点。同时，源结点从已经处理过查询请求的结点收到返回结果，接着执行步骤（3）。

（3）等待时间 W 后，如果源结点发现已经搜索到要找的内容，查询结果已经被满足，则执行步骤（8）。如果结果并不令人满意，则执行步骤（4）。

（4）源结点开始下一次迭代，进行深度为 b 的泛洪搜索。即，源结点发送 TTL=b 的查询请求，开始进行搜索。执行步骤（5）。

（5）进行如下的查询处理：

当源结点再次发送 TTL=b 的查询请求时，这意味着 a 跳距离内的每个结点需要重复处理查询请求。为了避免这一点，源结点发送 TTL=a 的 Resend 消息。a 跳距离内的结点不需要处理 Resend 请求，而只是进行转发。

当深度为 a 的结点收到 Resend 消息时，将 Resend 消息丢弃，同时“解冻”相应的查询请求，令查询请求的 TTL=b－a 向邻居结点发送。每个查询请求都有一个唯一的标识，Resend 消息里含有所代表的查询请求的标识，这样，前沿结点通过查看 Resend 消息里的标识就知道需要“解冻”哪个查询请求。执行下一步。

（6）进行深度为 b 的搜索之后，查询过程根据策略以同样的方式进行深度为 C 的迭代。由于 C 是策略的最后一次迭代的深度，查询在深度为 c 的结点处不会被“冰冻”。源结点

不会开始下一次迭代，即使查询结果仍然不令人满意。

（7）按照这种方式一直搜索下去，直到搜索到结果，或是直到执行步骤（8）。

（8）搜索过程结束。

以上的搜索过程如图 3.13 所示，显示的就是一个迭代递增的泛洪搜索过程。

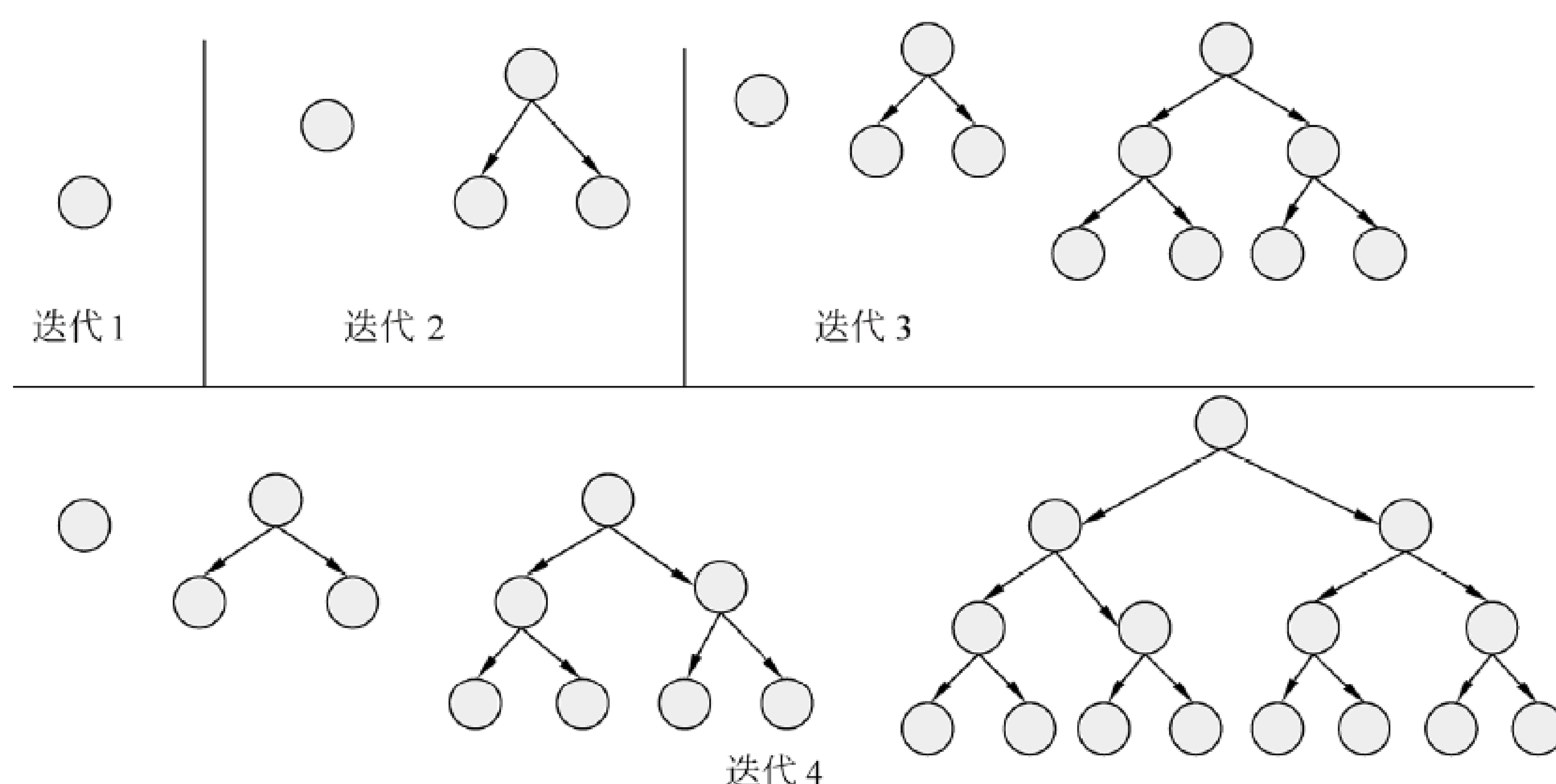


图 3.13　迭代递增的泛洪搜索

通过以上的迭代递增搜索，在好的情况下，迭代泛洪能够大量减少查询的结点个数，从而减少搜索的资源开销。因为迭代泛洪不一定要以 D 为深度泛洪，只要以比 D 小的深度泛洪就可以得到满意的查询结果。随着深度的增加，结点个数是按指数增加的。迭代泛洪避免了以大的深度进行泛洪，因此大量减少了查询的结点个数。

但是在坏的情况下，迭代泛洪可能要进行到最后一次迭代。这时，迭代泛洪就比直接泛洪方法更糟糕。因为多次迭代在网络上发送了大量的 Resend 消息，占用了网络带宽，也给结点增加了处理负担。

3.7.5　启发式泛洪搜索

泛洪搜索的另一种方式就是启发式泛洪，在启发式泛洪中，结点不是向所有邻居结点发送查询请求，而是选择其中一部分在过去表现优秀的邻居结点进行发送。这是基于这样一个假设：过去表现优秀的结点将来也会表现优秀，结点为了选择邻居，需要对每个邻居结点的信息进行统计。这些统计很简单，比如统计邻居结点在过去的查询中返回的结果个数，或者统计邻居结点在过去的查询中响应的时间等。基于这些统计结果，我们就能选择其中表现优秀的结点，一般的选择条件有以下几个：

- 选择在以往的查询中返回结果最多的邻居结点。
- 选择在以往的查询中响应时间最快的邻居结点。
- 选择在线时间最长的邻居结点。

通过这种方式，找到表现优秀的邻居结点后，查询请求就可以只发送给一小部分邻居结点，这样就能够大大减少查询的结点个数。另外，启发式泛洪选择的是以往表现优秀的

结点，减少的是以往表现不优秀的结点，因此查询的结果数量、查询的响应时间等在很大程度上能够得到保障。如图 3.14 展示的就是启发式搜索的模型。

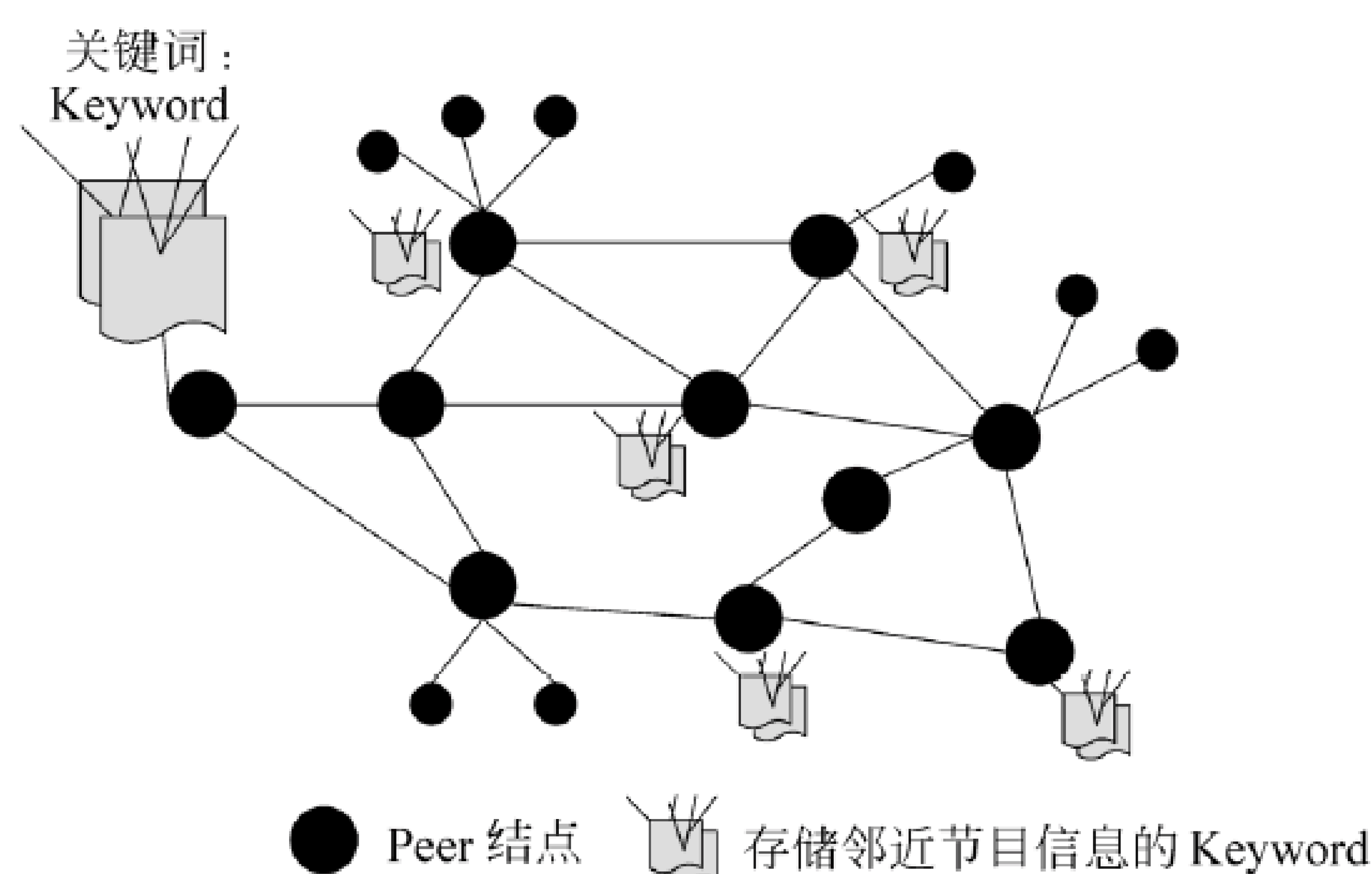


图 3.14　启发式搜索模型

从图 3.14 中可以看出，结点旁的一个 KeyWords，记录了邻结点的统计信息，这样就便于查询的转发和结点的选择。

启发式泛洪假设过去表现优秀的结点将来也会表现优秀，但是，这个假设不是永远都成立的。而且，如果资源恰恰存放在过去表现不好的结点上，采用启发式泛洪就会搜索不到资源，因为启发式泛洪只会搜索表现优秀的结点，而不是搜索每个结点。

3.7.6　Random Walk 搜索方法

Random Walk 搜索也叫随机漫步搜索，在这种搜索方法中，请求者发出 K 个查询请求给随机挑选的 K 个相邻结点。然后每个查询信息在以后的漫步过程中直接与请求者保持联系，询问是否还要继续下一步。如果请求者同意继续漫步，则又开始随机选择下一步漫步的结点；否则中止搜索。如图 3.15 展示的是随机漫步的效果图。

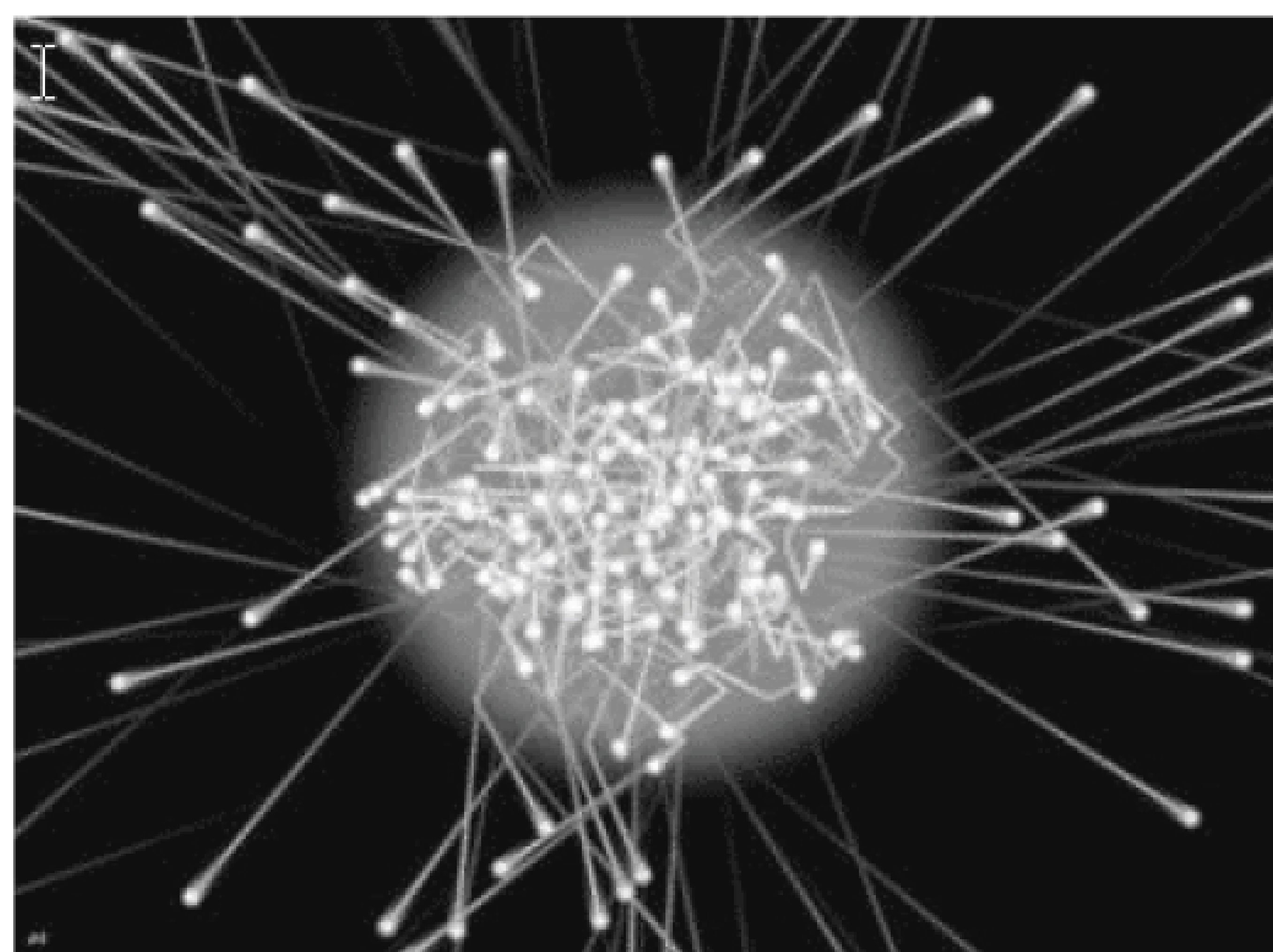


图 3.15　随机漫步效果图

随机漫步搜索也是对 Flooding 算法的一种改进，同前面的改进算法相比，这种算法对

结点路由消息的控制更加明显，在路由消息的扩散范围和扩散程度两个方面都对算法进行了改进。

使用该算法进行路由时，首先查询结点将随机地挑选 n 个相邻结点发送查询请求，中间结点每次随机地选择自己的一个相邻结点转发消息，查询请求在路由过程中与请求结点保持联系。查询请求也被称为 walker。查询请求在以下 3 种情况下停止继续路由。

- 成功找到查询的内容；
- 当消息的 TTL 为 0 时；
- 定期地发送 checking 消息给查询结点，询问是否终止查询。

在 Random Walk 路由算法中，两个参数的取值对算法性能有很大的影响，一个是 Walker 的数量 n，另一个是 checking 消息的步长 S。如果 n 取值太大，将增大网络的负载，如果 n 取值太小，将增大查询的延时，一般来说 n 的取值为 16～64；如果 S 取值太大，将降低验证作用，增大网络的负载，如果 S 取值太小，将造成查询结点出现阻塞。

3.7.7　小世界模型（Small World）对 P2P 搜索技术的影响

小世界模型也称六度空间理论，就是说，如果用有向图模拟社会网络，任何两个结点之间最多只要用 6 条边就可连接。换句话说，你和任何一个陌生人之间所间隔的人不会超过 6 个。

小世界概念描述的是这样的意思，这个世界上不认识的两个人之间，如果通过他们的朋友关系建立连接的话，那么这条朋友链的长度不超过 6。这种现象就叫做小世界。用数学的概念来描述就是：无向图中所有结点对的最短路径边数的平均值称为图的直径，又称为特征路径。

大量的试验表明，一条成功链的平均中间步数是在 5 步到 6 步之间。在通信网络中，各个结点之间都是以链条的形式连接在一起的，如果网络中的任何两个结点都能以短跳距相连，即可以通过少量的链路往返在网络中定位存储于任何随机结点中的信息，这个网络就具备了小世界模型的一般特征。小世界现象是大量网络在自然和技术上所呈现的一个特征，在 P2P 网络显的更加突出。

注意： 关于小世界现象，在 1976 年的时候，哈佛大学教授 Milgram 作了这样一个试验，在内布拉斯加州的奥马哈随机选出 160 个当地居民，让他们每人设法将一封信传递到麻省波士顿的一个零售店主手中。这些人将信交给他认为最接近目标的一位朋友手中，接下来由这位朋友以同样的方式传递到下一位朋友手中，直到最后信送达目标人手中。实验结果是，最后有 42 封信到达了目标人手中，而且这些信的平均中间传递人数为 5.5 人。相比较 2 亿美国人口，这个数字显得非常低。这就是非常著名的“小世界”现象。

小世界模型的特性是网络拓扑具有高聚集度和短链的特性。在符合 Small World 特性的网络模型中，可以根据结点的聚集度将结点划分为若干簇（Cluster），在每个簇中至少存在一个度最高的结点为中心结点。大量研究证明了以 Gnutella 为代表的 P2P 网络符合 Small World 特征，也就是网络中存在大量高连通结点，部分结点之间存在“短链”现象。图 3.16 展示了网络重叠的 Small World 现象。

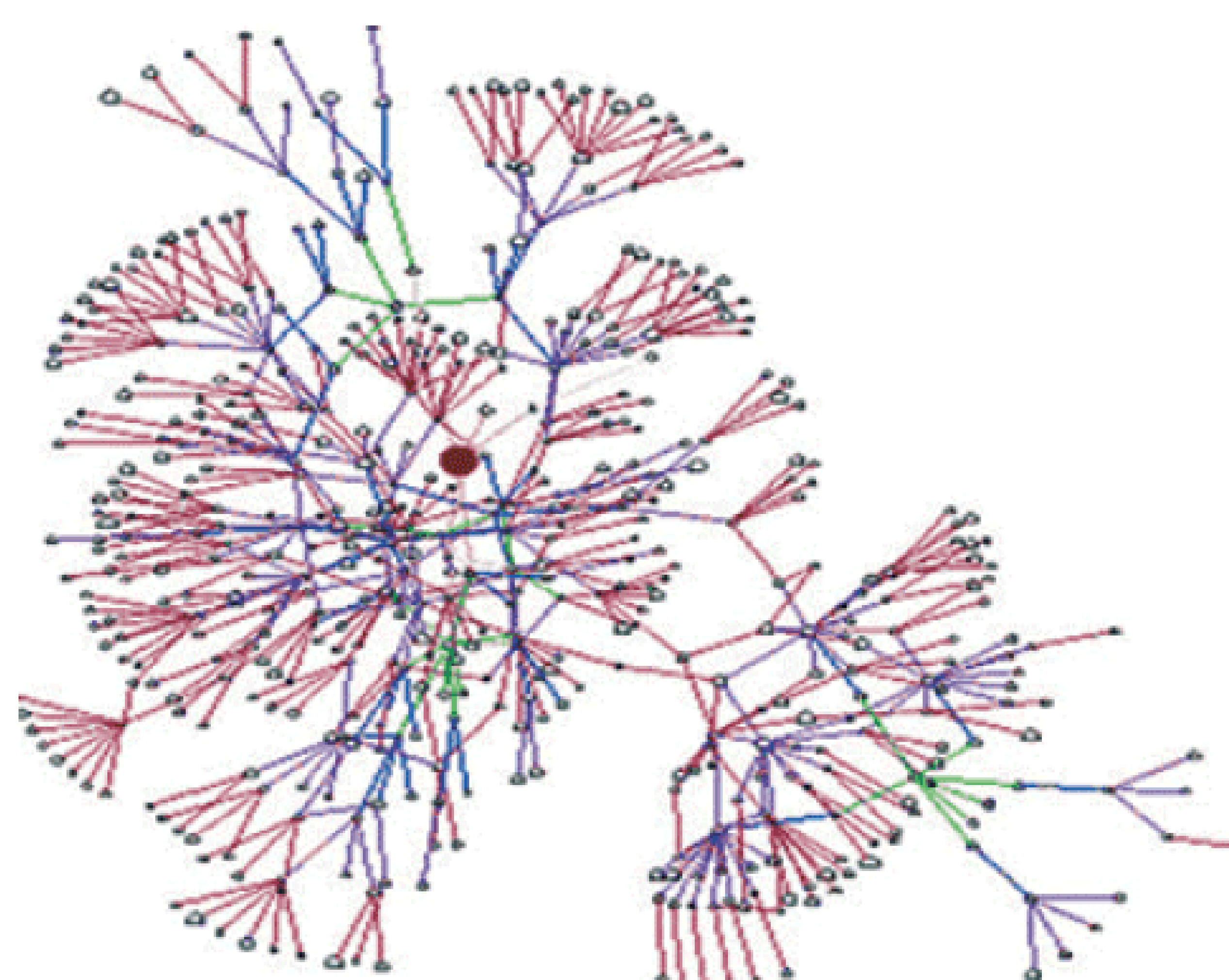


图 3.16　网络重叠的 Small World 现象

因此，P2P 搜索算法中如何缩短路径长度的问题变成了如何找到这些“短链”的问题。尤其是在 DHT 搜索算法中，如何产生和找到“短链”是搜索算法设计的一个新的思路。Small World 特征的发现和引入会对 P2P 搜索算法产生重大影响。

3.8　混合式 P2P 网络搜索技术

这里所说的混合式 P2P 搜索技术，并不是基于混合式 P2P 结构的搜索技术，而是在同一 P2P 搜索过程中，运用了两种或两种以上的搜索技术进行混合搜索的方法。从某种意义上说，它并不是一种特有的技术，而是几种搜索技术的有效混合，搭配合作的方法。因为，已存在的搜索算法多种多样，各有优缺点。如果将这些算法综合起来，或许可以得到较好的结果。

3.8.1　混合的鸡尾酒搜索算法

鸡尾酒搜索（cocktail Search）算法，主要是考虑到目前没有任何一个单一的搜索算法在所有方面都是完美的，因此希望通过混合这些已有的搜索算法来提高搜索性能。其思想是每个结点同时有几种搜索算法供选择，当消息到达时，会根据当时的情况，概率地选择合适的搜索算法来转发消息，避免单一算法的缺陷。这个思想与著名的艾滋病鸡尾酒疗法相似。

注意： 鸡尾酒搜索算法，来源于鸡尾酒的说法，鸡尾酒指的两种或两种以上的酒和果汁、香料等混合而成的酒，后来就用这个意思指将多种因素综合起来的一种应用。

3.8.2　模拟退火思想的混合搜索算法

为了探索随机行走搜索和泛洪搜索之间的不确定的性能空间，获得系统的性能提升，有学者提出了一种基于模拟退火思想的混合搜索算法，在搜索开始的时候，即搜索时间低

的情况下进行洪泛搜索。这里引入参数 n 来说明，在最初开始的 n 个搜索步里，搜索空间应该是全局的，转发查询消息给所有的结点。当搜索时间大于 n 时，算法就局部搜索网络，即采用随机行走搜索。这种方式可以综合利用泛洪搜索和随机行走搜索的优点，从而有效地克服了它们的不足。

注意：模拟退火是一种通用概率算法，用来在一个大的搜寻空间内找寻命题的最优解的一种算法思想。它来源于自冶金学的专有名词退火，退火是将材料加热后再经特定速率冷却，目的是增大晶粒的体积，并且减少晶格中的缺陷。对于其他知识有兴趣的读者可自行查阅相关资料。

3.8.3　Gnutella 2 的搜索算法

Gnutella 2 是对原有 Gnutella 算法的一种改进，它在网络的结构中建立了 Super-Node，超级结点，Gnutella 2 的超级结点图如图 3.17 所示。

这些超级结点存储着离它最近的叶子结点的文件信息，同时相互之间再连通起来形成一个 Overlay Network。与混合式的 P2P 网络结构类似，当叶子结点需要查询文件，它首先从它连接的 Super Node 的索引中寻找，如果找到了文件，则直接根据文件所存储的机器的 IP 地址建立连接，如果没有找到，则 Super Node 把这个查询请求发给它连接的其他超级结点，直到得到想要的资源。KaZaa、POCO 等都是基于这种超级结点的思想。

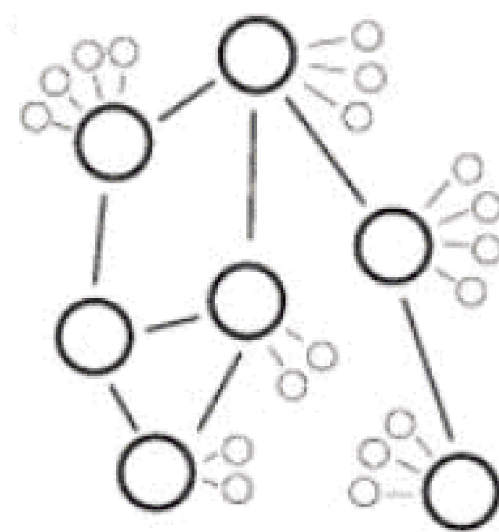


图 3.17　Gnutella 2 的 Super Node 结点图

3.9　P2P 搜索技术研究的机遇与挑战

P2P 搜索技术作为 P2P 技术体系中的核心技术之一，在整个 P2P 发展的历程中始终扮演着重要的角色。在当前 P2P 发展的大潮中，作为 P2P 重要技术之一的搜索技术，不仅面临着重要的发展机遇，也遇到了严峻的挑战。

3.9.1　P2P 搜索研究的机遇

随着 P2P 系统实际应用的发展，P2P 搜索技术正面临着巨大的发展机遇。由于 P2P 网络蕴含着巨大的技术潜力和商业价值，许多学术机构和大公司先后投入到对 P2P 技术的研究之中。P2P 系统的流行和成功推动了关于不同技术方面的客观研究，这是 P2P 技术不断发展的原动力。

随着网络硬件的不断发展，结点的服务能力、存储能力及带宽容量都得到的极大的提高，这能有效地弥补 P2P 在搜索方面的先天不足，充分发挥其优势。

新的 P2P 网络模型和搜索算法正在深入研究中，原有的搜索技术也在不断的发展和完

善中，这些都给 P2P 搜索带来了新的发展机遇。

3.9.2　P2P 搜索技术面临的挑战

P2P 搜索技术中最重要的研究成果应该是基于 Small World 理论的非结构化搜索算法和基于 DHT 的结构化搜索算法。但是 P2P 在实际的应用中，以下因素影响了搜索的效率。

1．网络结点异质性

物理网络中影响路由的一些因素开始影响 P2P 发现算法的效率。由于 P2P 客户机/服务器模式在 Internet 和分布式领域十几年的应用和大量种类的电子设备的普及，如手提电脑、移动电话或 PDA，这些设备在计算能力、存储空间和电池容量上差别很大。在通信基础方面，P2P 必须提供在现有硬件逻辑和底层通信协议上的端到端定位（寻址）和握手技术，建立稳定的连接。实际网络被路由器和交换机分割成不同的自治区域，体现出严密的层次性。

2．网络的波动程度

网络波动（Churn）包括结点的加入、退出、失败、迁移、并发加入过程、网络分割等。网络的波动的影响结点发现算法的效率。

3．复查查询的需求

对复杂的查询，如关键词、内容查询等，DHT 精确关键词映射的特性阻碍了 DHT 在复杂查询方面的应用。P2P 搜索方法一直是研究的热点，但是与传统的搜索技术还有很大的差距。研究搜索速度快、成本低、分布性好、支持多关键词搜索、带宽要求小的 P2P 搜索方法，可以使搜索技术大步向前迈进，走出搜索形式多样化的道路。

3.9.3　P2P 搜索的进一步研究方向

近年来，工业学术界从体系结构、信息查找、路由机制、系统安全等方面对 P2P 系统实现涉及的几个关键技术开展了工作。由于对等计算尚处于发展初级阶段，数据管理、结点组织、信息检索等方面还有许多问题有待解决。

1．覆盖网络重组，降低网络流量

当前的 P2P 应用占用带宽已超过整个网络流量的 60%，成为 Internet 的“杀手级”应用，而且随着应用普及还有加大的趋势。这是因为一方面是计费模式的问题，而更重要的一方面是 P2P 数据组织、搜索定位等技术的不完善，尤其是 DHT 往往全局命名，忽略物理网络结构，造成带宽的“浪费”。P2P 系统最大的优点之一在于充分发挥网络边缘计算能力的作用，进一步工作应该完善覆盖网络构造过程中的物理邻近性问题。

2．属性描述的需求

对数据实现基于自然语言处理技术、本体技术、语义技术的自动分类和聚合，进一步

实现个性化搜索、领域化搜索、语义推理智能化搜索，从而提高搜索性能和服务质量。在进一步工作中可以借鉴语义 Web 技术，实现对数据、服务的语义描述，扩展知识表示语言，丰富上下文环境，加强基于日志和语义的自动推理，实现对 P2P 数据和服务的自动查找、发现、匹配和整合。

3．异构融合与协议标准

P2P 系统充分整合网络边缘资源，所以 P2P 系统一个很大的特点是“强之欲强，弱之越弱”。各个独立的数据共享系统之间形成信息孤岛，不能共享。而且，当前 P2P 领域至今尚没有基本的成型的协议标准。进一步工作应该注重系统之间的互通接口，加强系统之间的资源整合，加强系统之间的信息和数据共享。

总之，当前的 P2P 搜索研究已经不仅仅停留在可行性研究阶段，而是以专业化、个性化、智能化为目标，综合带宽节约、负载均衡等性能要求，提高成功率、响应时间方面的服务质量需求。这些问题的解答不仅对提高 P2P 系统性能、改善网络带宽利用具有重要的现实意义，而且对当前 web service 中的资源发布、普适计算中的资源管理等也具有极其重要的参考借鉴价值。

3.10　本 章 小 结

P2P 搜索由于其独特的优势，得到了快速的发展，为了能够更深入地理解 P2P 搜索技术，本章在 P2P 网络拓扑结构的基础上重点讲解了 P2P 网络中的主要搜索技术，并说明了结构化网络和非结构化网络中的一些常见搜索方法。在结构化网络中，每个结点存储的信息与网络拓扑结构有关，通过映射完成，查找采用基于 DHT 分布式散列路由搜索算法。而非结构化网络则与网络拓扑无关，其结点可任意存储信息，查找采用基于广度优先的泛洪搜索算法及其改进算法。这两种不同结构的网络所采取的搜索技术是完全不同的，在学习本章的时候需要重点掌握。

多年来，P2P 搜索算法一直是人们研究的热点，各种搜索算法纷纷被人们提出来，使得 P2P 搜索日益趋向成熟，它的各种优势也日益体现出来。随着基于 P2P 的应用逐渐成为 Internet 的主要消费者，P2P 搜索技术必定有更加广阔的应用和发展空间。

第 4 章　P2P 的关键技术及其应用

在学界有这样一句话：“理论是应用发展的基础，理论是为应用服务的工具”。离开了应用，为了理论而理论，就会言之无物，演变成空中楼阁，而没有理论指引的应用，就会迷失方向，演变成低水平的重复。只有“理论——技术——应用”的推进，才能真正地实现用技术和理论去指导应用并最终为社会、为人们的生活服务的目的。

P2P 也是一样，在 P2P 发展的过程中，形成了很多经典的理论和关键的技术，这些技术是 P2P 得以存在和发展的基础，也是 P2P 真正能够为人类服务的前提，本章将主要讲述 P2P 的关键技术及其典型应用的相关知识，从整体上对 P2P 涉及的技术和日常生活中出现的 P2P 应用进行说明，以使读者对 P2P 技术体系和应用现状有整体的、较全面的理解。本章涉及的知识点如下。

- P2P 涉及的主要技术：理解 P2P 所涵盖的主要技术体系，如体系结构技术、内容存储技术、内容查询技术及 P2P 的传输技术等。重点要理解技术原理及要点。
- P2P 技术的典型应用：理解 P2P 技术在当前互联网中的典型应用，尤其在文件共享与下载、内容分发、分布式计算及通信与协作等方面的应用，同时了解基于这些应用的分类及代表性的平台和软件。
- P2P 技术的企业级应用：了解 P2P 技术在企业级的应用，了解 P2P 对资源生产、资源传输、资源交互方面的影响，以及对互联网的价值体现。
- P2P 应用带来的问题：了解 P2P 技术带来的问题，理解产生这些问题的原因。

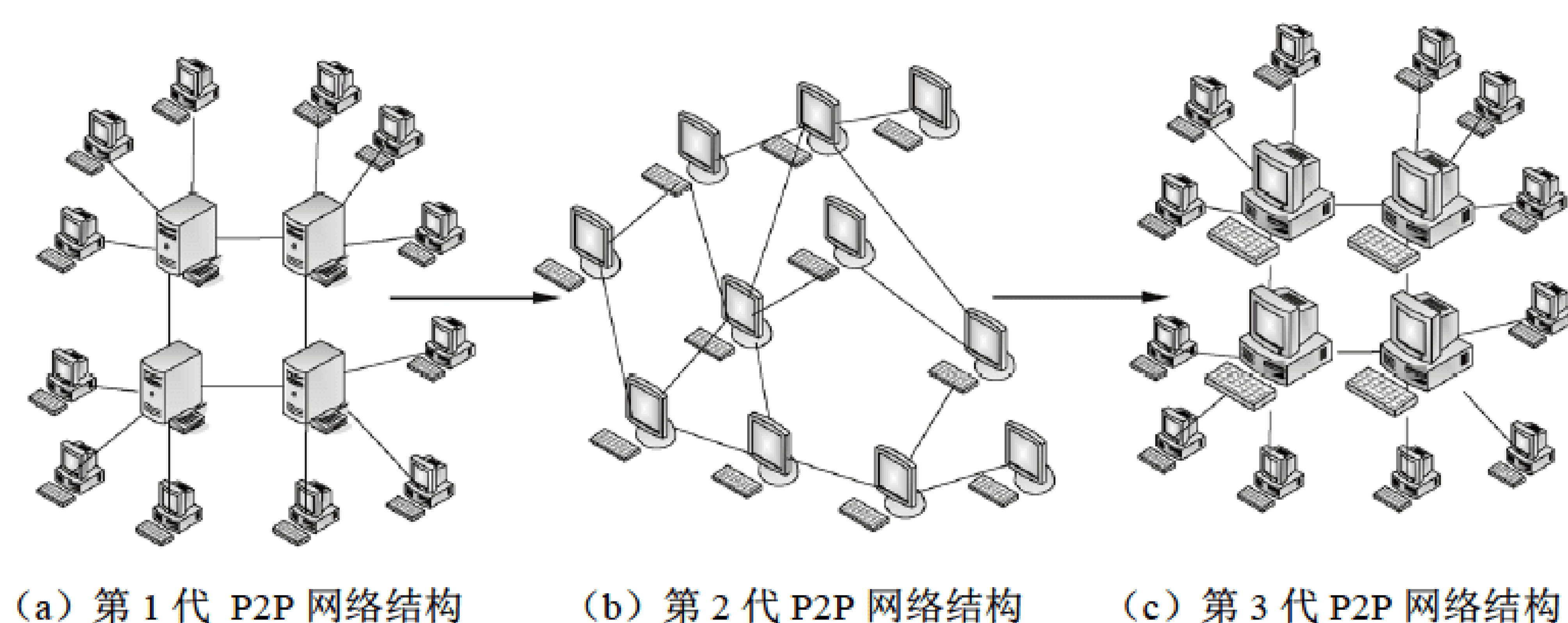
4.1　P2P 的体系结构技术

P2P 并不是天生存在的，它是一种架构在互联网上的虚拟重叠网络，依赖于独特的计算机互联体系架构，它将成千上万的计算机用户连接起来，彼此提供和共享资源与服务。它的形成、组织及其存在的结构都需要相应的技术规范来实现，可以说，P2P 的体系结构是其存在并发挥作用的前提，要研究 P2P 的相关技术，就先从 P2P 的体系结构说起。本节就重点讲一下 P2P 的体系结构所涉及的技术。

4.1.1　动态的变化

P2P 的体系结构根据其不同的特征，有第一代、第二代、第三代之分，每一代均有不同的特点，如图 4.1 所示，是三代 P2P 网络结构的演变与特征。但在每一代的结构里面，都有一个共同的特性，就是动态性。也就是说任何一个 P2P 网络都是动态变化的，因为，对任何一个成型的 P2P 系统而言，组成这种 P2P 关系的主机没有数量、范围、时间或空间

上的限制，其中每一台主机都可以随时、自由地加入或退出，这就表现出 P2P 结构动态的特征。



（a）第 1 代 P2P 网络结构　　（b）第 2 代 P2P 网络结构　　（c）第 3 代 P2P 网络结构

图 4.1　P2P 的体系结构分类及演变

举一个简单的例子，通过 eMule 文件共享系统下载文件这一过程，来简单说明一下 P2P 网络的动态特征。

（1）状态 1：当前没有什么文件需要用 eMule 下载，尽管电脑上装有 eMule 软件，但没有启动它。在这种状态下，计算机只是互联网中千万台计算机中的一员，不属于任何 P2P 网络。

（2）状态 2：有一个文档，需要用 eMule 才能下载下来，于是启动了 eMule。在这种状态下，一旦使用 eMule 软件开始进行资源的交互了，那么，这台主机就成为 P2P 网络中的一员了，此时，这个主机就是一个 P2P 的网络结点，是一个活跃的 Peer。

（3）状态 3：文件下载完成后，退出了 eMule 程序，在这种状态下，主机脱离了 P2P 网络，又成为互联网中普普通通的一员，不再与 eMule 网络发生任何关系，也不与其他 P2P 网络中的任何结点发生交互。这样，就正式地退出了 P2P 网络。

注意：这种假定是在计算机中除了 eMule 以外，没有安装其他任何的基于 P2P 协议的应用程序。

在整个动态的 P2P 网络体系结构中，除了某些特殊的服务器以外，几乎每个结点都会发生由状态 1 到状态 3 的变化，也就都会经历从加入 P2P 网络——运行于 P2P 网络——到退出 P2P 网络这样一个变化过程，每时每刻会不断的有新用户加入进来，也不断的有老用户离开。这个过程在 P2P 网络的整个生命周期内都是在不停地进行着。

因此，面对这样一个结构，在 P2P 系统中都需要引入一种动态成员的管理机制，专门进行用户加入、离开以及出错的管理，这是一个应用型的 P2P 系统必须要解决的技术问题。

例如在 CAN 系统中，整个系统在逻辑上是一个 d 维的 Cartesian 坐标空间，每一个用户分别管理一个小型的矩形坐标区域。当有新用户加入时，需要有一个老用户将自己的管理空间分成两半，将其中的一半分给新用户；一个用户离开时，他的管辖区域要递交给另一个用户等。伴随着管辖区域变化的，还有每个用户所需保存的一些成员信息的变化等。这些都是对 P2P 网络中的结点进行管理的良好机制，当然还有其他系统中对 Peer 结点管理的其他方法，总之，P2P 系统的动态性及其基于动态的管理机制，是 P2P 研究的重要技术。

4.1.2　平等的参与

P2P 的思想就在于对等和共享，对 P2P 网络中的任意两个结点而言，如何定义这种平等，如何保证它们之间的平等，如何让这种平等在动态的 P2P 网络中延续下去，以及如何控制这种平等不被破坏等，都是 P2P 需要解决的问题。

在 P2P 系统中，需要正确地理解对等的概念，这里所说的平等，指的是平等地参与，平等地享有权利与义务，以及在网络中的地位及性质平等。并不是说，在用 eMule 的时候，下载了 100MB 的文件，就必须上传 100MB 的文件才算平等。实际上指的是，在用 eMule 的时候，当下载时，作为使用者，你在使用其他的 Peer 给你提供的服务，那么同时，你也在上传资源，这个时候作为服务者，你也在给另外的 Peer 提供服务。在这种参与、提供或消费服务上，双方的地位及性质是对等的，这才是 P2P 系统中对等的意思。如图 4.2 所示，在 P2P 网络中的主机，每个主机都是一个对等的结点，在地位、关系、角色上都是对等的。

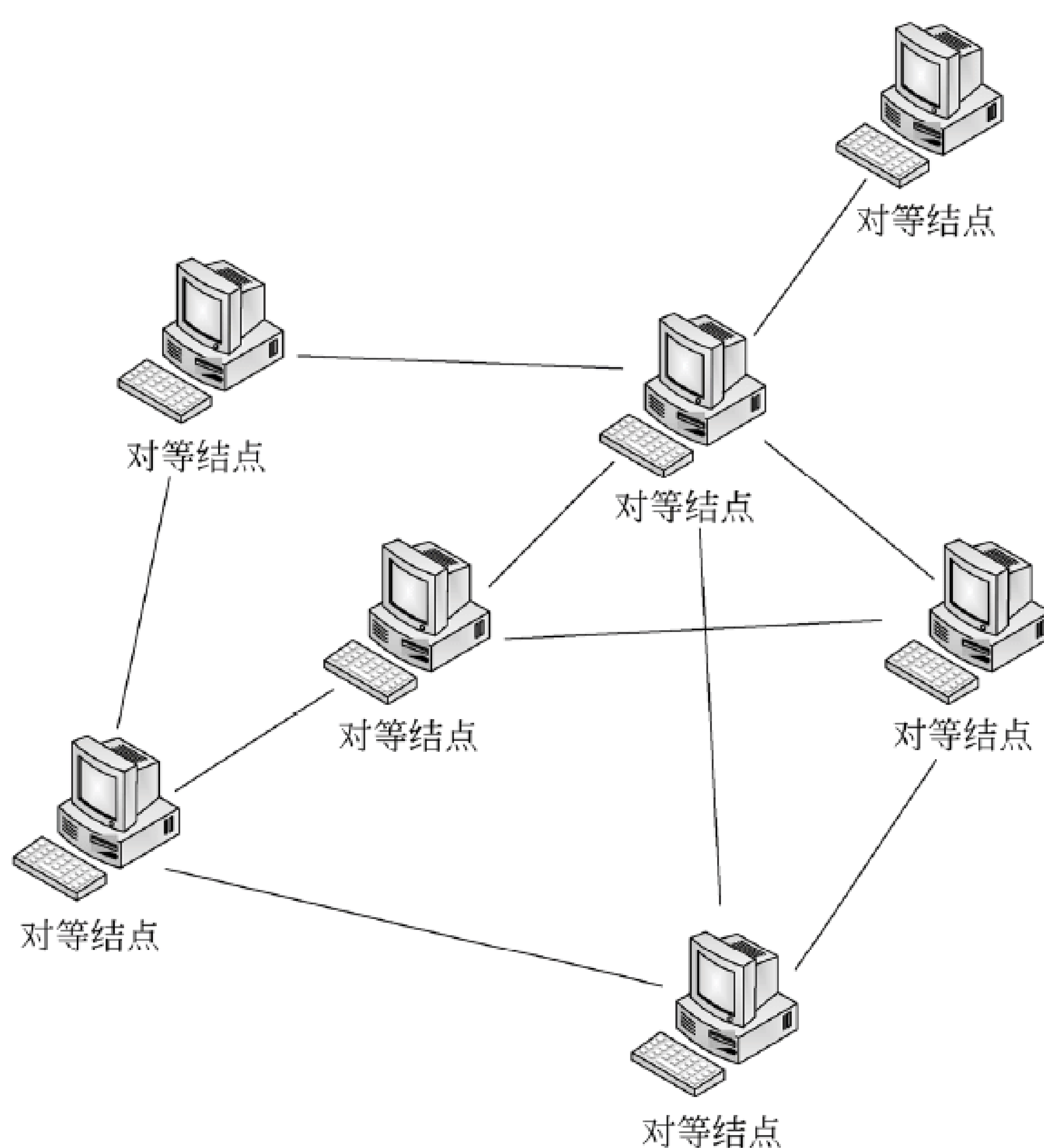


图 4.2　P2P 网络中的对等结点

任何一个 P2P 系统，都需要保证每一个 Peer 都是平等的参与者，承担服务使用者和服务提供者两个角色。保证资源的所有权和控制权不被集中在某一台或几台主机上，而是被分散到网络的每一个结点中。服务使用者和服务提供者之间不经过中间服务而直接进行通信。

P2P 系统中，要实现结点之间平等的参与，还要解决系统的伸缩性问题，同时也要避免单点故障，提高系统的容错性能。

4.1.3　分散与自治

P2P 系统是动态的，Peer 结点是自由的，也就决定了其参与结点的分散与自治特性。在 P2P 的体系结构技术中，必须要解决结点的分散性与自治性问题。

分散性很好理解，分散是与集中相对应的，指的就是组成 P2P 网络的结点没有统一的组织规律，在时间和空间上都是极其分散的，它的分散性表现在以下几点。

（1）在时间上的分散性，单个结点来说，它可以在任意时间内加入 P2P 网络，也可以随时退出。对整体的结点来说，各个国家、地区的时区分布不一样，人们的上网习惯也不一样，结点的加入和退出也随着改变，因而从时间上看，是极其分散的。

（2）在空间上的分散性，P2P 网络是架构在 Internet 上的网络，Internet 能走到哪，P2P 网络的结点就有可能出现在哪。Internet 网络已经互联到世界上的每个角落，那么加入到 P2P 网络中的结点就有可能出现在世界的任何地方，因而其在空间上的分布也是极其分散的。

除了分散性这一特点，另一显著的特征就是 P2P 网络的自治性，它的自治性指的是 P2P 网络中的 Peer 结点在没有中央控制器的前提下，它的运行、控制和管理都靠自身来进行。

P2P 网络体系结构必须满足结点分散与自治的特性，同时也由于 P2P 网络的分散性、自治性、动态性等特点，造成了某些情况下 Peer 的访问结果是不可预见的。例如，一个请求可能得不到任何应答消息的反馈，这些问题都是 P2P 技术需要解决的，解决了这些问题才能真正实现一个可靠性好、稳定性好，能够应付爆发式访问的健壮的 P2P 网络系统。

4.1.4　网络结点中的角色划分

在 P2P 网络体系结构中，还需要解决的另一问题就是，网络结点中的角色划分。前文已经讲过，在 P2P 网络中，有两种典型的拓扑结构，即纯 P2P 网络和混杂的 P2P 网络。在纯 P2P 网络中，每个 Peer 都具有同等的责任和地位，不存在中间结点的协调。FreeNet、Gnutella 都属于纯 P2P 网络。而在混杂的 P2P 网络中，存在着充当服务器角色的 Peer 结点或提供特殊功能的 Super-Peer 结点，这些结点保存其他 Peer 结点的基本状态和存储内容源信息，协助完成对其他结点的记录、查询等工作。Napster、Groove、Magi 等系统均是典型的混杂型 P2P 系统。那么，这就有一个问题，要设计一个 P2P 系统，如何区分这些结点的不同功能，如何划分它们的角色呢？

在一个 P2P 系统中，每一个 Peer 根据其提供的角色功能可以划分为 3 种类型，即简单类型 Peer 结点，超级结点（super-peer）和服务器型的 peer 结点。如图 4.3 所示，分散在外围的就是简单的 Peer 结点，而处于中心位置担当类似服务器角色的结点，就是服务器型的 Peer 结点。

简单类型 Peer 结点仅仅是一个简单的终端用户，它可以请求获得服务并为网络中的其他 Peer 提供服务。

Super-peer 结点除了具有和简单 Peer 结点类似的功能外，还提供某些特殊功能。例如 JXTA 系统中就存在路由器 Peer 结点和会聚点 Peer 结点，前者作为一个桥梁，使得被防火墙或 NAT 隔离的 Peer 可以相互通信；后者为简单结点提供查询定位信息的功能。

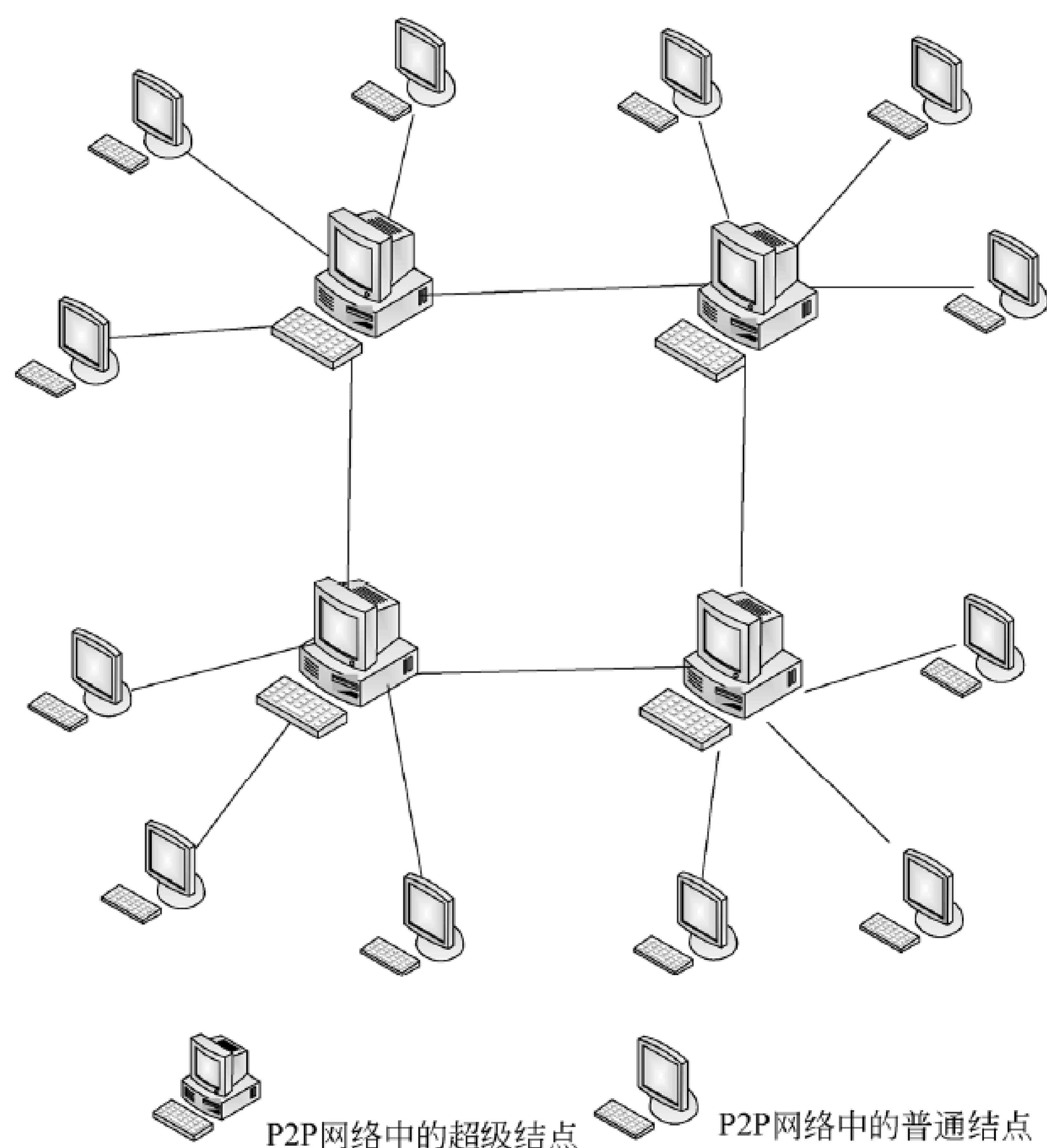


图 4.3　P2P 网络中不同角色的结点

服务器型的 Peer 结点主要提供类似于客户/服务器模型下的服务器功能，如提供一个全局统一的目录查询。在 Napster 这种混杂型的 P2P 系统中，有若干个简单 Peer 结点相互提供文件下载功能的服务，还有一个目录服务器结点提供文件条目的注册管理。Groove 和 Magi 系统中也均存在这样的服务器型结点。而在纯 P2P 网络中，每一个 Peer 结点均充当了上述的 3 种角色。

根据不同的应用需求，设计 P2P 系统的时候，就需要考虑 Peer 结点的角色划分，需要解决 P2P 体系结构中结点的角色划分问题，这样整个系统一旦运行起来后，不同的结点才能做到各司其职，各负其责，让整个 P2P 系统高效地运转起来。

4.2　P2P 的内容存储技术

P2P 系统使用户间可以彼此平等地共享资源，这些资源并不是集中存储在某一个服务器上，而是分散地存储在各个 Peer 结点中，而且资源的类型包罗万象，应有尽有。那么，如何标识这些资源的存在？如何获取这些资源？这些都是 P2P 内容存储技术需要解决的主要问题，本节就重点讲解一下在 P2P 系统中关于内容存储的相关技术。

4.2.1　资源的标识

在日常生活中，如果我们要找一个人，首先就会找这个人的名字，如果没有他的名字，

就会找与他相关的一些特征信息，如年龄、职业、住址等。名字及其他相关信息综合起来可以称为这个人的标识信息，因为这些信息表征了这个人的存在。在 P2P 网络中也是一样，资源是分散地存储在各个结点上，在时间和空间上都没有规律可循。要想找到某一个资源，其前提就是赋予资源一个特定的标识，找到了这个标识就找到了资源。

在 P2P 网络系统中，为了在 P2P 网络中准确地查找资源进行 Peer 定位，就需要确定 Peer 中存贮资源的标识（这里我们将在 P2P 网络中需要进行查找的对象统称为资源）。不同的应用场景均有适合自身特点的资源标识方式。

在以文件共享为主的应用中，资源主要以文件的名称、关键字、源数据等进行标识。而即时消息通信系统往往采用类似于电子邮件的命名方式。如在 Jabber 系统中，Jabber 的用户 ID 以[node@]domain/[resource]的形式进行地址标识，提供一个全局统一的地址空间。如图 4.4 所示，是 Jabber 的运行界面图，从图中可以清楚地看到在 Jabber 中 ID 的标识形式。

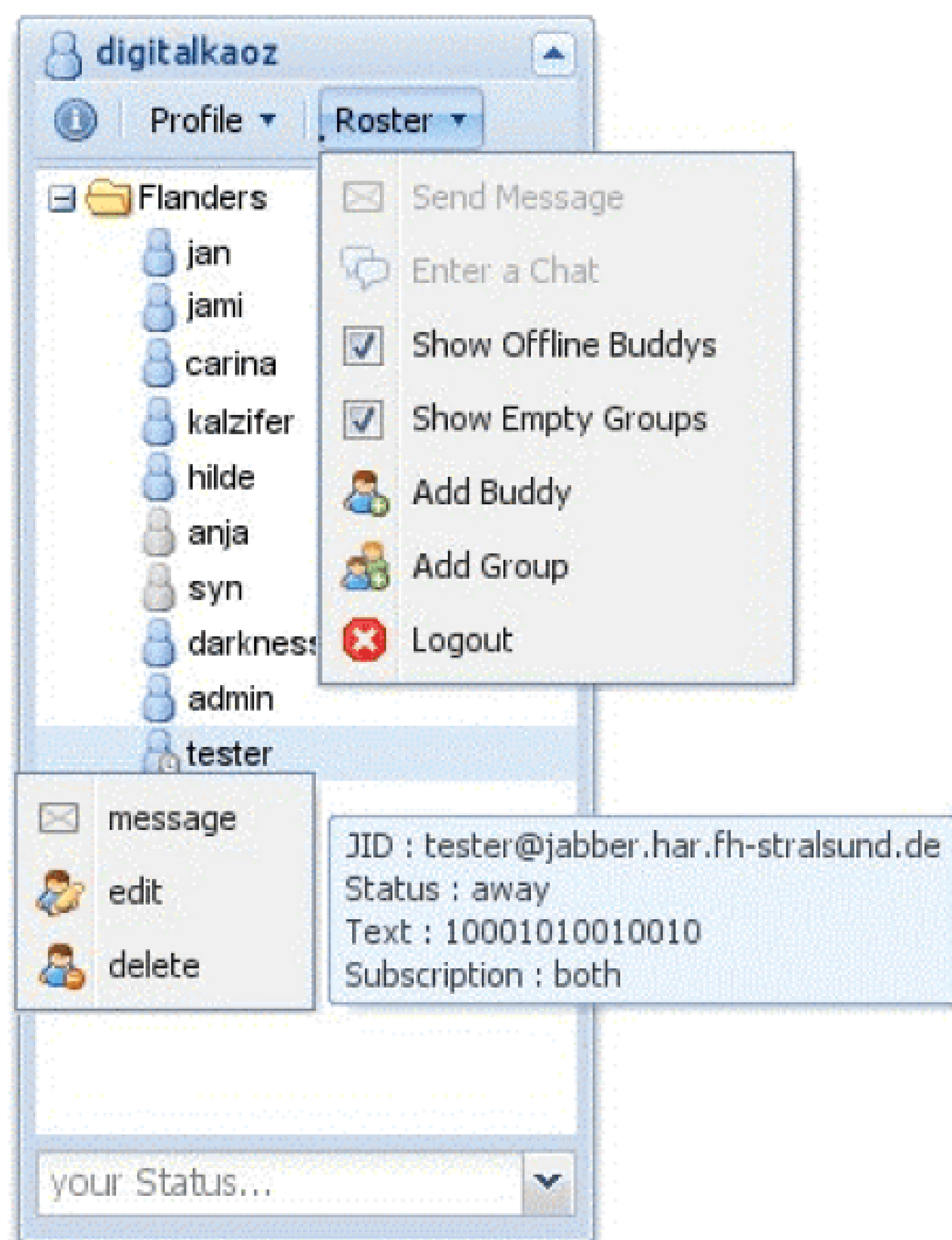


图 4.4　Jabber 中 ID 的标识形式

其中，Domain 是主要的 ID 标识，是与多个用户连接进行消息转发的 Jabber Server；Node 为用户姓名或昵称，Resource 属于一个 Node，标识属于一个用户的多个资源。一个用户可以同时与同一服务器建立多次连接。

注意： Jabber 是著名的 Linux 即时通信服务服务器，它是一个自由开源软件，能让用户自己构架即时通信服务器，可以在 Internet 上应用，也可以在局域网中应用。Jabber 最有优势的就是其通信协议，可以和多种即时通信对接。比如有第三方插件，能让 Jabber 用户和 MSN 、Yahoo Messager、ICQ 等 IM 用户相互通信。关于 Jabber 的相关知识，在后文还会讲到。

以用户之间协作为主的 P2P 应用如 Groove 系统和 Magi 系统为例，由于在协作中对即

时消息通信和文件共享的要求兼而有之，用户一般以用户名进行标识，而文件则以源数据的特征来标识。

JXTA 作为一个解决不同 P2P 应用彼此不兼容的底层平台，提出了一个统一的资源标识定义，即广告（advertisement）。广告是一个 XML（Extensible Markup Language，可扩展标记语言）结构的文档，由 Peer 或相互协作共同完成一种应用 Peer 组进行发布，用来描述现有的资源、服务或实体 Peer。Peer 结点通过查询广告以获得各种资源的消息，进行 Peer 定位。

注意：关于 JXTA 平台的知识，在后文有专门的章节讲解。

基于 P2P 的应用系统有很多，不同的应用服务、不同的需求特征对资源也有不同的标识方法，但不管怎么说，在任何一个 P2P 系统中，对每一个参与共享的资源都得给它一个标识，这是查找并定位 Peer 的前提。

4.2.2　资源的获取

P2P 系统中资源的数量数以万计，要准确地查找并定位 Peer 中的资源其前提是要给资源一个标识，但这只是做完了第一步，还需要解决资源的获取问题。P2P 中的资源是个广义的概念，包括一切用于在 P2P 网络中共享的对象，如硬件、软件、计算能力等，资源的获取技术，就是提高 Peer 结点间访问成功率、提高资源的可获得性的技术。

P2P 系统使用户间可以彼此共享资源，为提高资源的可获得性，很多 P2P 系统都采取了复制和缓存技术。复制是将文件复制保存在离请求发起用户距离较近的用户结点中，缓存一般是把有关的历史查询信息存储在结点上，为以后的查询提供参考信息。

例如，在 Napster 系统中，用户可以访问其他用户计算机上的 MP3 文件。为了提高这种访问成功率，即提高资源的可获得性，就可以使用复制和缓存的技术，利用复制（Replication）将文件复制保存在离请求发起用户距离较近的用户结点中。当用户发起请求的时候，只需经过较少的查找就能找到所需的资源；而缓存（Caching）则有多种不同的策略，在不同的 P2P 系统中使用的缓存技术也不一样。例如在 FreeNet 系统中，文件在发布过程中，每经过的一个用户，文件都将被复制保存在该用户中；文件找到后，在传向请求发起用户的返回路径上，经过的每个用户也将保存该文件副本。复制和缓存都可以减少查询文件所经过的路径长度，从而减少用户间的消息传输数量，降低通信延迟。同时，当系统中的一个用户出现问题时，保存在其他用户上的冗余信息能够使系统正常运行。

资源的标识以及资源的获取可以说都是 P2P 系统中关于共享内容存储的一种方案，其根本目的就是要把尽可能多的资源有效地纳入到 P2P 系统的统一管理范围内，最大限度地实现所有参与 Peer 结点之间高效的交互与共享。

4.3　内容查询技术

P2P 系统中，一个用户要共享另一个用户计算机上的资源，不论是文件、存储空间或是计算资源，一个关键的问题是要找到资源所在的目的主机，找到指定的需求资源，因此，

内容的查询是很多 P2P 系统的核心，也是 P2P 研究的重要技术之一。在前面的章节中，专门对 P2P 的搜索技术进行了详细的讲解，搜索技术也是 P2P 在内容查询方面研究内容之一，本节总体上对 P2P 的内容查询技术进行说明。

4.3.1　Peer 的定位方式

P2P 网络中的所有资源都是存储在各个分散的 Peer 结点上的，要对其中的某一资源进行查找，其本质就是查找存储着这一资源的 Peer，因而，P2P 中内容查找的过程就是 Peer 定位的过程。在 P2P 网络中 Peer 的定位方式可分为以下几类，它们的分类关系如图 4.5 所示。

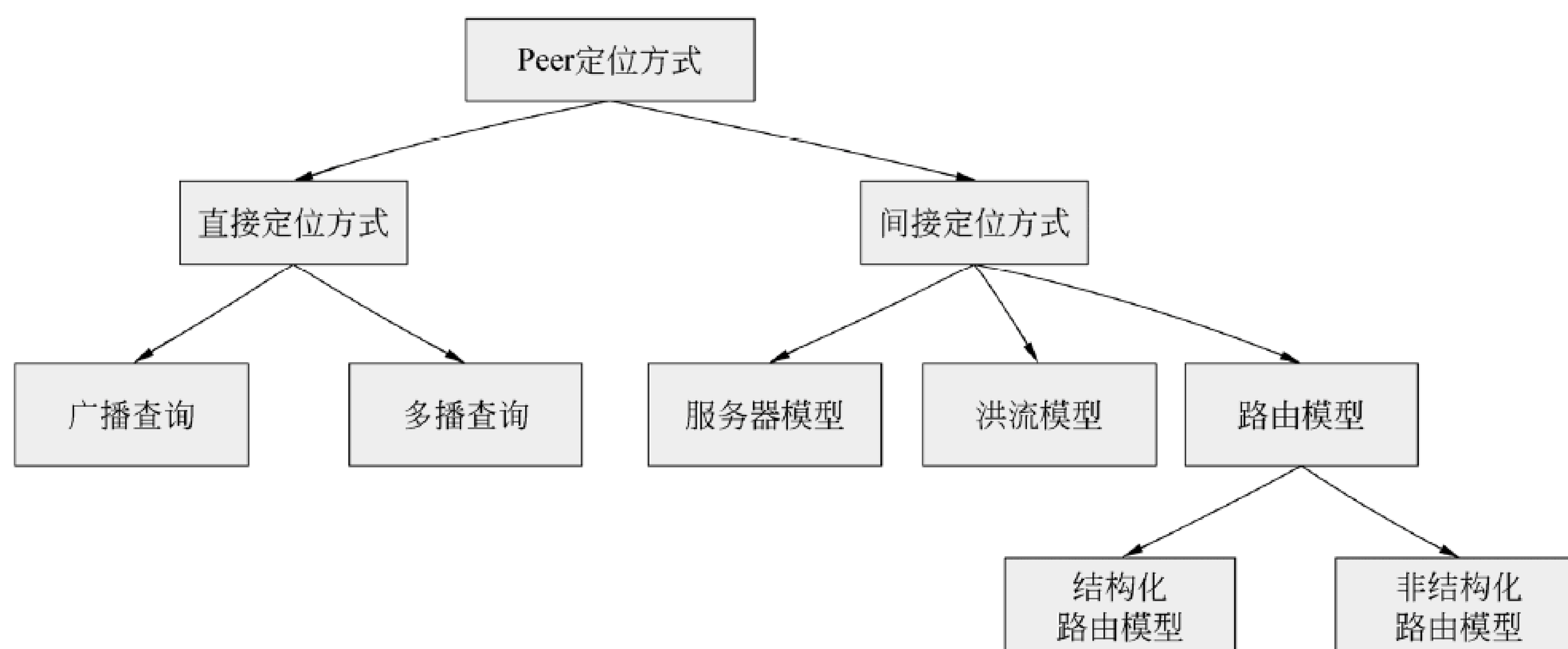


图 4.5　Peer 定位的分类

在 P2P 网络中，查找资源的时候一般可采用直接或间接方式来定位 Peer。直接定位 Peer 的方式比较简单，即利用广播或多播的形式发出查询请求，符合查询要求的 Peer 结点进行应答，然后建立直接的通信连接。如图 4.6 所示为网络中进行多播的示意图。

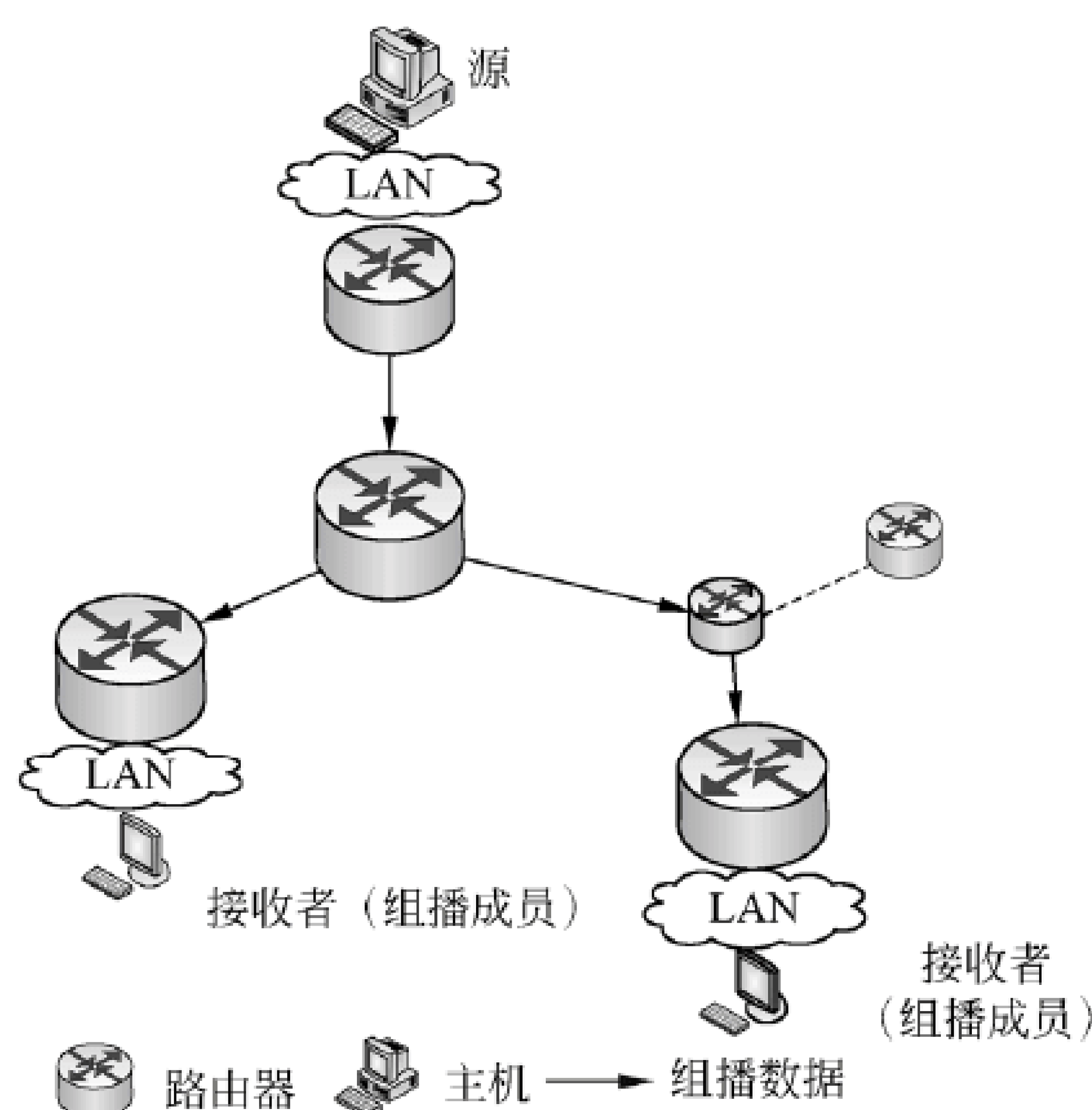


图 4.6　多播示意图

由于这种方式只能在局域网中使用，所以应用范围有限。当然这种方式可以和其他的定位方式结合使用以获得良好的查询效率。

注意：多播（multicast），在一些文献中也被称为多点传输、多点广播、群播、组播等，都是同一个意思，是指把信息同时传递给一组目的地址。它使用策略是最高效的，因为消息在每条网络链路上只需传递一次，而且只有在链路分叉时，消息才会被复制。与多播相对应的另一个概念就是单播，常规的单点到单点的传递就是单播。

间接的 Peer 定位方式应用比较广，包括 3 种定位模型：服务器模型、洪流模型、路由模型。

1．服务器模型

该模型是基于混杂型的 P2P 拓扑结构。充当服务器的 peer 结点提供资源查询。peer 将请求发送至服务器获得查询结果，随后，直接与目标结点通信获取所需服务。但这种方式存在单点失败问题，同时，也存在伸缩性问题。因为 Peer 结点仅在启动、停止及查询的时候才与服务器交互，所以此时的伸缩性还是强于客户/服务器模式。典型系统有 Napster、Magi、Groove 和 Jabber。Napster 系统采用一个目录服务器记录资源索引，Groove 系统和 Magi 系统均是采用动态 DNS 查找用户的 IP 地址。Jabber 系统由于其资源标识类似于 E-mail 地址，所以，利用 DNS 的 MX 记录进行 IP 地址的查询。

2．洪流模型

该模型基于纯 P2P 拓扑结构。Peer 结点采用洪流法将查询请求不断地转发至邻居结点，直到到达目标结点，获得查询结果。同时为了避免消息无限制地转发，查询请求中设定有 TTL（Time to Live）或 HTL（Hops to Live）进行转发控制。Gnutella 是采用此类模型的典型系统。

3．路由模型

该模型也是基于纯 P2P 网络结构。首先为网络中的每一个 peer 赋予一个 ID，同时，每个 Peer 存储的资源和服务也有类似的 ID。Peer 结点的路由表中登记一定数量的邻居结点。Peer 的请求被转发至与所请求的资源或服务的 ID 最接近的 Peer，直到发现这个资源或服务。插入一个新资源/服务的过程与查询过程类似，也是通过查找该资源/服务 ID 来确定存储的正确位置。此类模型主要用在文件共享系统中，如 FreeNet、Chord、CAN、Tapestry、Pastry 等。

路由模型又可细分为非结构化路由模型和结构化路由模型。FreeNet 系统属于典型的非结构化路由模型。在查找到所需资源后，为了提高搜索性能，系统沿搜索路径复制资源。这样，由于资源的存储位置不固定，其行为不易观察，不确定因素较大。所以相对于结构化路由模型来说，其资源分布的规律性不强，难以从全局上把握整个系统的资源分布状况。而结构化路由模型如 Chord、CAN、Tapestry、Pastry 均采用了 DHT（Distributed Hash Table）作为主要的存储算法。DHT 的主要思想是将资源定位用的索引（索引结构通常是两元组（文件名，实际的存储位置））分散存储到整个 P2P 网络上，这样，哈希表的存储和查询操作就会涉及到 P2P 网络中的多个结点。

以 DHT 思想进行路由模型的设计时，首先需要确定通过 Hash 函数进行虚拟地址空间映射的规则。虚拟地址空间的设计有多种方式，Chord 采用的虚拟地址空间为 m-位的循环地址空间，CAN 系统采用的是多维的地址空间。Peer 结点的 IP 地址和端口号经过哈希函数映射到地址空间，再将映射空间进行划分，每个结点负责存储属于自己空间的值对（key、value）。其次需要确定路由表项的存储内容，即邻居结点的规模，以适应于不同的网络需求。

这里需要对路由表项的规模与网络搜索跳转数进行综合考虑。在动态性较强的网络中，结点频繁加入和退出网络会使得规模较大的路由表更新频率过高，操作费时。但规模较小的路由表在进行资源定位时，又使得查找时间过长。再次是确定在接收到一个资源的查询请求时，从路由表中选择转发的邻居结点的规则。最后要确定新结点的插入和删除操作后，虚拟的地址空间如何进行分裂和合并。

4.3.2　搜索算法

Peer 的定位方式是对结点搜索和内容查询的大致描述，而要实现这种方式就需要相应的搜索算法来完成。图 4.7 展示了 P2P 网络中 Peer 的一次搜索过程，图中展示的搜索信息从一个结点传到另一个结点，直到找到目的结点。从定位方式上来讲，它是一种间接的基于洪流模型的定位方式，但从实现的角度来讲，它有一个详细的搜索算法来规定这个搜索的过程。

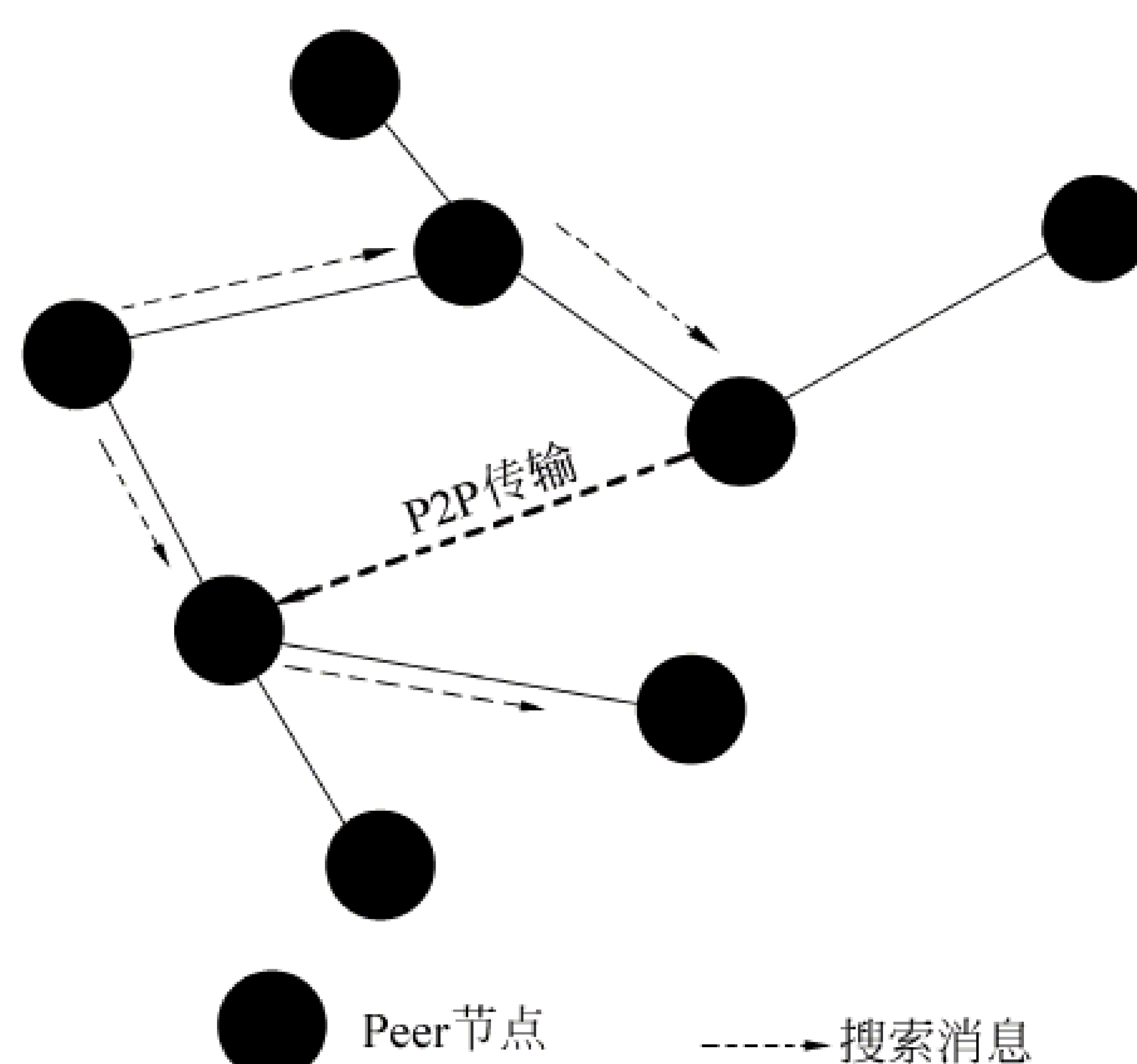


图 4.7　P2P 网络中 Peer 的搜索过程

简单描述这个过程就是：每个结点在加入网络的时候，会对存储在本结点上的内容进行索引，以满足本地内容检索的目的。然后按某种预定的规则选择一些结点作为自己的邻居，加入到 P2P 网络中。发起者 P 提出检索请求 q，并将 q 发送给自己的邻居，P 的邻居收到 q 后，检查本身是否存在查询的信息，如果不存在，转发查询，直到返回结果。

以上这一过程就需要相应的搜索算法来实现，第 3 章专门讲解了 P2P 的搜索技术，对其中的搜索算法有详细的讲解，这里再对 P2P 的搜索算法进一步总结说明。概括起来说，P2P 搜索算法可以分为以下 3 种。

1．集中索引算法（Central-index）

此算法主要以 Napster 系统为代表。在 Napster 系统中，用户都与一个中央服务器相连

接，中央服务器上保存了共享文件的索引。由中央服务器对收到的用户请求进行匹配查找，直到找到保存了所需文件的目的用户。然后，由发起请求的用户与目的用户直接进行文件交换。这种算法的不足在于依赖一个集中式的结构，将会影响系统的可扩展性。

2．泛洪消息算法（Flooded requests）

泛洪消息算法的典型代表系统为 Gnutella。每一个用户消息都将被广播给与该用户直接相连的若干其他用户，这些用户收到消息后，也同样地将消息广播给各自连接的用户，依次类推，直到请求被应答，消息的 TTL 值减少为 0 或超过了最大的广播次数（通常在 5 和 9 之间）。这种算法存在的不足在于需要较大的网络带宽，因此也会影响可扩展性。

3．文件路由算法（Document routing）

文件路由算法的代表系统是 FreeNet。算法的特点是采用基于哈希函数的映射，也就是分布式哈希表映射。在这种搜索算法中，系统中的每一个用户都有一个随机的 ID 序列号；系统中的每一个文件也有一个 ID 序列号，这个序列号是根据文件的内容和它的名字，经过哈希函数映射得来的。文件发布时，每一个用户都把文件转发到拥有与文件 ID 最相近的 ID 值用户上，直到最接近文件 ID 的用户就是该用户本身。转发过程中经过的每一个用户都将保持该文件的副本。索取文件时，每个用户都将请求消息转发给一个拥有与所需文件 ID 最相近的 ID 的用户，直到文件或文件的一个备份被发现为止。Tapestry、Pastry、Chord，CAN 都是采用这种方法的 P2P 系统。这种算法的优势在于可扩展性较好，不足在于它可能导致整个网络分裂成若干彼此不相连的子网络，形成所谓的孤岛，它的查询也要比洪水消息等算法麻烦些。

4.3.3　精确及深度检索

P2P 技术的另一个优势是开发出强大的搜索工具。P2P 技术使用户能够深度搜索文档，而且这种搜索无须通过 Web 服务器，也可以不受信息文档格式和宿主设备的限制，可达到传统目录式搜索引擎（只能搜索到 20%～30%的网络资源）无可比拟的深度（理论上将包括网络上的所有开放的信息资源）。

以 P2P 技术发展的先锋 Gnutella 进行的搜索为例，一台计算机上的 Gnutella 软件可将用户的搜索请求同时发给网络上另外 10 台计算机。如果搜索请求未得到满足，这 10 台计算机中的每一台都会把该搜索请求转发给另外 10 台计算机，这样，搜索范围将在几秒钟内以几何级数增长，几分钟内就可搜遍几百万台计算机上的信息资源。可以说，P2P 为互联网的信息搜索提供了全新的解决之道。著名的搜索引擎公司 Google 也宣称要采用 P2P 技术来改进其搜索引擎。关于 P2P 的精确及深度检索技术，也是未来 P2P 研究的重点。

4.3.4　关于结点的登录和退出

在 P2P 的内容查询技术中，还需要注意的一个问题，就是结点的登入和退出，从 P2P 网络的结构上讲，任何 Peer 在任何时间都可以自由地加入或退出 P2P 网络，那么 Peer 结点在加入或退出 P2P 网络时其本身就是一次 Peer 的定位过程，这一过程可从如下几个方面

来说明。

在服务器模型的 P2P 网络中，由于 Peer 结点的状态信息和管理的资源信息直接记录在服务器中，所以 Peer 结点的登录和退出需要和服务器进行交互。当你登录的时候，需要告诉服务器你的位置信息，退出的时候服务器也要记录你的状态，这些都由服务器进行协调处理。如在 Napster 系统中，Peer 结点登录系统后，向服务器发送当前的网络状态和所有的文件列表信息，由服务器更新目录索引。在 Jabber 等即时消息通信系统中，Peer 结点往往是与用户绑定的。服务器接收到用户的登录消息或退出消息后，会通知订阅该用户在线状态的所有在线用户。

在非结构化的 P2P 路由模型中，如 FreeNet 网络，新结点首先需要获知网络中的若干可连接结点的信息，通过执行搜索操作向整个网络宣布自己的存在。

在结构化路由模型的 P2P 网络中，结点的登录和退出，将导致虚拟地址空间的分裂和合并。Peer 结点登录网络的操作，类似于一个资源的查找过程。Peer 结点定位到所属的地址空间后，将地址空间以一定规则进行分裂。负责管理该地址空间资源的原结点将所属资源按分裂的规则，部分转移到新结点。同时两个结点的路由表项和其他相关结点的路由表项均要进行修改。而 Peer 结点退出网络的过程，则是登录网络的逆过程，资源需要转移合并，相关结点的路由表项同样需要更新。

4.4　内容传输技术

要实现 P2P 网络的对等共享和资源的交互，一个重要的问题就是内容的传输问题。当两个结点之间建立连接以后，如何把资源从一端传输到另一端呢？如果多个结点同时交互，还要保证交互的正确和高效，还要克服防火墙问题、传输质量及传输的数据类型等多种问题，这些都是 P2P 内容传输技术需要解决的问题。

4.4.1　P2P 通信

P2P 通信时需要解决的最基本的问题即是如何连接其他的终端获得信息、资源和服务。但这一通信过程并不是孤立的，需要在 Peer 结点的功能角色划分、资源和服务的标识及 Peer 定位的基础上进行；还需要解决如何穿越 NAT（Network Address Translation）和防火墙的问题，以进行 Peer 结点之间的直接通信。

P2P 通信另一个重要的问题就是建立稳定的连接。消息在传输过程中，对于互联网上众多的计算机，P2P 应用比其他应用要更多考虑那些低端 PC 的互联，因为它们不具备服务器那样强的联网能力。同时对于以往的 P2P 应用技术，现在的硬件环境已经更为复杂，因此，P2P 必须提供在现有硬件逻辑和底层通信协议上的端到端定位（寻址）和握手技术，建立稳定的连接。

在 P2P 通信中研究的其他技术还有 IP 地址解析、NAT 路由及防火墙等。此外，在应用层方面，如果两个用户通过互联网建立连接，那么一方的信息就必须为另一方所识别，所以 P2P 系统还需要包含关于数据描述和交换的协议，如 XML、SOAP、UDDI 等。

注意： 以上提到的 XML、SOAP、UDDI 等，是 SOA（面向服务架构）里用到的一些重要概念，其中 XML 标准是一个基于文本的 World Wide Web 组织（W3C）规范的标记语言。SOAP 简单对象访问协议（Simple Object Access Protocol）是一个基于 XML 的，用于在分布式环境下交换信息的轻量级协议。而 UDDI、统一描述、发现和集成（Universal Description, Discovery and Integration）规范提供了一组公用的 SOAP API，使得服务代理得以实现。

4.4.2　互操作性

P2P 的用户千差万别，在系统平台、软硬件信息、主机性能及版本控制上都不尽相同，这就需要一个统一数据描述和交换的协议。也就是说，P2P 在内容传输与通信的过程中需要解决互操作性问题。

举个例子，P2P 网络中有两个对等的 Peer 结点，比如它们分别代表两家不同的公司，一个是中国的，一个是德国的。中国的公司不理解德语，而德国的公司也不理解中文，当它们之间要进行交流的时候就必须借助第三方的、双方都可识别的语言来进行，那么这个双方都可识别的语言，就是这两家公司进行互操作的共同平台。

P2P 网络中的结点也与此类似。两个对等的 Peer 结点之间可能它们的差异相当大，当它们直接通信时可能相互都无法识别，那么一方的信息就需借助一个通用的平台发布出去，另一方也通过这个平台识别出信息，这一过程就是 P2P 网络中关于数据描述和交换的过程。在当前的技术条件中，有很多可以实现这种互操作的协议，如 XML、SOAP、UDDI 等。要设计一个完善的 P2P 软件，就要尽可能地考虑互操作性。

4.4.3　防火墙和 NAT 的穿越

在实际的网络通信中，Peer 结点往往是一个私有网络中的结点，位于防火墙之后。这样，Peer 与 Peer 之间直接通信需要解决的一个关键问题是穿越防火墙和 NAT。

由于防火墙会对 IP 进行过滤，限制了防火墙内外的连接，而 NAT 技术虽然可以使得内部网络地址映射到外部网络地址，但要求内部网络首先发起对外连接，否则外部网络机器无法达到内部网络。穿越防火墙和 NAT 就成了 P2P 通信中一个必备的技术。图 4.8 为 P2P 系统中 NAT 穿越的示意图。

关于 NAT 技术及穿越防火墙的问题在本书的第 5 章有详细的讲解，这里只对基本的穿越策略作简要说明。穿越防火墙的策略有两个基本点。

- 关于协议：需要使用在一般情况下可以通过防火墙的协议，如基于请求/应答方式的 HTTP 协议。
- 连接的发起者：Peer 之间进行通信时，必须由内部网络的机器首先发起连接请求，如果通信双方均处于防火墙之后，则需要有防火墙外的转发结点进行消息转发。

在 Napster 系统中当 Peer 结点请求下载的资源位于一个防火墙之后的结点上，请求结点向服务器端发送需要进行转换下载的请求（Alternate download request），由服务器通知被请求结点首先发起文件下载的连接请求。

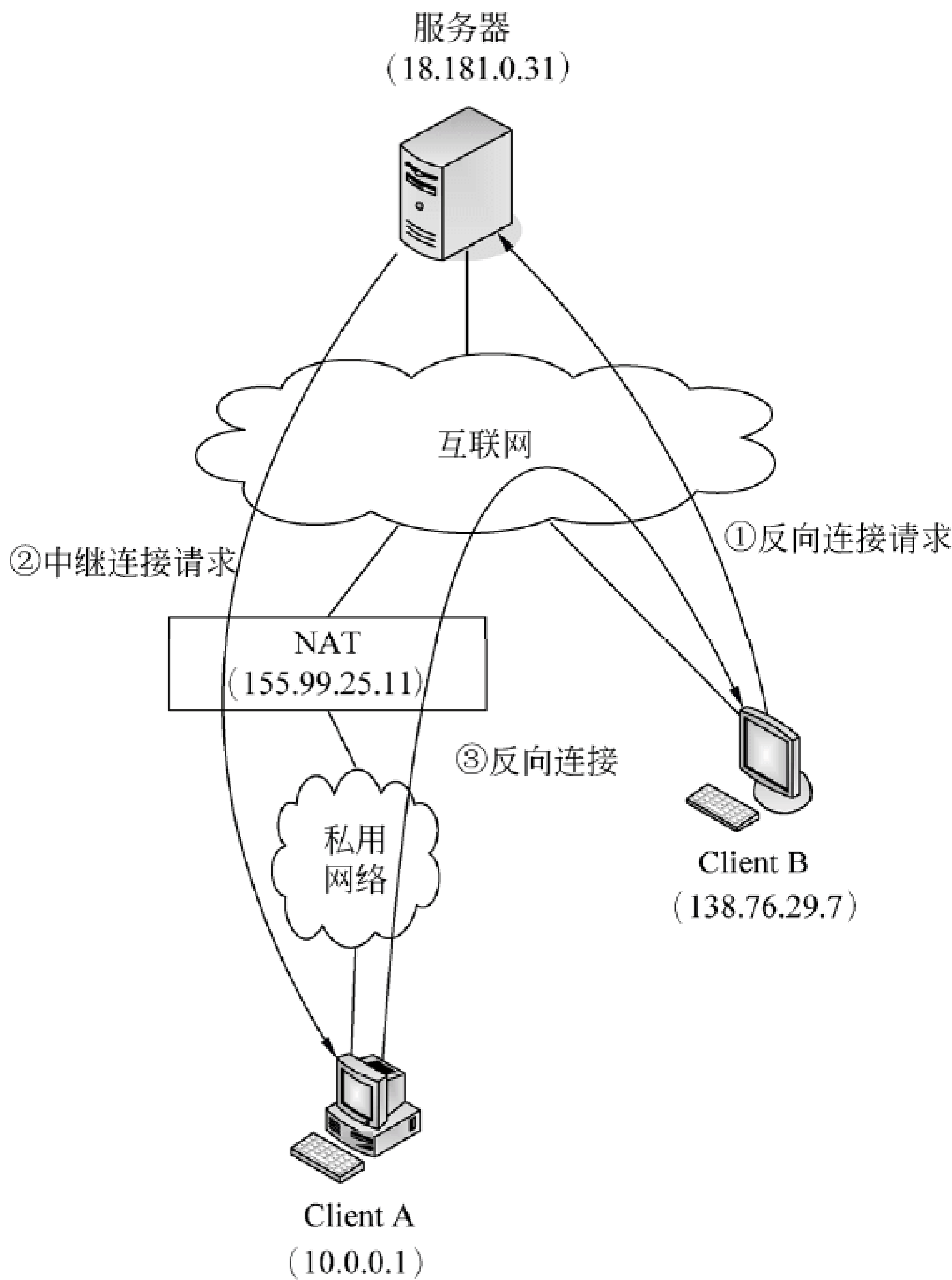


图 4.8　P2P 系统中 NAT 穿越示意图

而 Gnutella 是在将查询的详细信息返回给请求者的同时发送资源文件，以此方式解决文件提供者位于防火墙之后的情况。

注意：这部分的详细内容请参阅第 5 章。

4.4.4　网络服务质量（QoS）问题

QoS 的英文全称为“Quality of Service”，中文名为“服务质量”。QoS 是网络的一种安全机制，用来解决网络延迟和阻塞等问题的一种技术。

在正常情况下，如果网络只用于特定的无时间限制的应用系统，并不需要 QoS，比如 Web 应用，或 E-mail 设置等。但是对关键应用和多媒体应用就十分必要。当网络过载或拥塞时，QoS 能确保重要业务量不受延迟或丢弃，同时保证网络的高效运行。

P2P 网络的 QOS 问题包括两个方面。

- 信息获得的 QOS 问题，用户需要的信息在多个结点同时存放，如何选择一个处理能力强、负载轻、带宽高的结点需要用户考虑。
- 用户共享出无用或者违法信息，造成信息垃圾充斥网络，因此，网络应该控制用户共享的信息，提高用户获得有用信息的效率。

在 P2P 网络中，每时每刻都有大量的数据在进行交互传输，网络带宽占用多、数据流

量大，这就必然导致 P2P 网络中存在网络延迟和阻塞的问题。如果没有一种机制来解决这个问题，那 P2P 系统就没有什么实用价值。

要解决网络延迟和阻塞的问题除了 QOS 技术之外，还有其他一些关键问题，如 P2P 网络标准、资源共享版权、现有网络带宽等，都是需要考虑到的。

4.4.5　P2P 流媒体传输

在 P2P 的内容传输中，还需要解决一类特殊数据的传输，就是流式数据传输的问题，P2P 中的流式数据就是通常所说的流媒体信息。当前基于 P2P 的流媒体应用十分火爆，发展前景也异常广阔，本书有专门的章节来讲述 P2P 流媒体技术，这里先简要地提一下。

所谓流媒体是指采用流式传输的方式在 Internet 播放的媒体格式。流媒体又叫流式媒体，它是特定的流媒体传送服务器把节目当成数据包发出，传送到网络上。用户通过解压设备对这些数据进行解压后，节目就会像发送前那样显示出来。

在传统的基于 Web 的多媒体应用中，流媒体数据一般是基于客户/服务器模式进行发送的，服务器以单播的方式和每个用户建立连接。由于流媒体服务具有高宽带、持续时间长等特点，所以随着用户数量的增加，服务器的带宽很快被消耗完，经常出现断点、不断缓冲甚至服务器崩溃等问题。而应用 P2P 技术进行流媒体传输则可以很好地解决这一问题。

由于 P2P 网络本身的可扩展性，基于 P2P 方式的流媒体技术很好地解决了传统流媒体带宽不足的问题。单源的 P2P 流媒体系统建立在应用层组播技术的基础之上，由一个发送者向多个接收者发送数据，接收者有且只有一个数据源。服务器和所有客户结点组织成组播树，组播树的中间结点接受来自父结点组播的媒体数据，同时将数据以组播的方式传送给子结点如图 4.9 所示，单源的 P2P 流媒体模型。

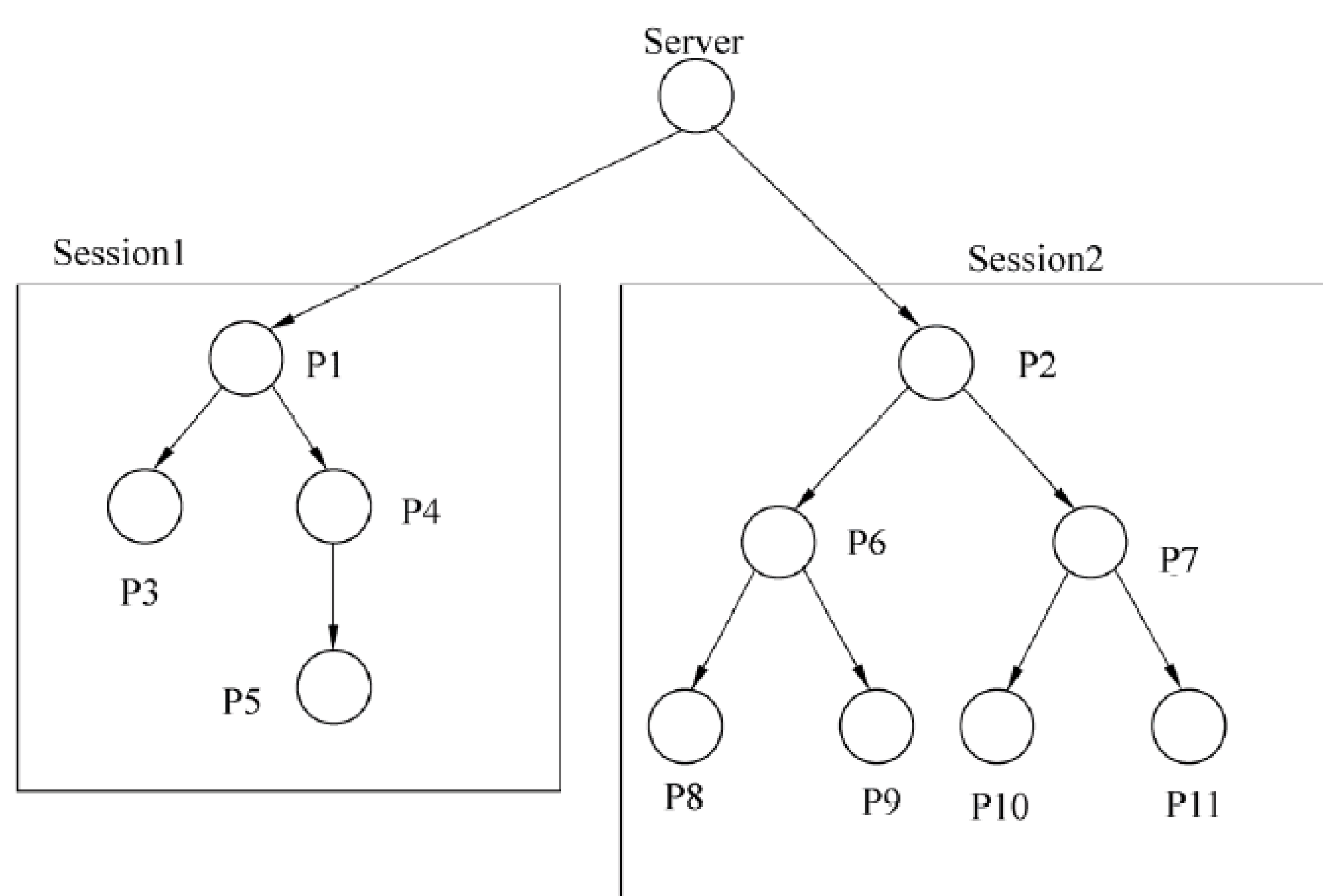


图 4.9　单源的 P2P 流媒体模型

而多源的 P2P 流媒体传输系统，则是由多个发送者以单播的方式同时向一个接收者发送媒体数据，这样就能有效避免服务器的过度负载、如图 4.10 所示，是多源的 P2P 流媒体传输模型。

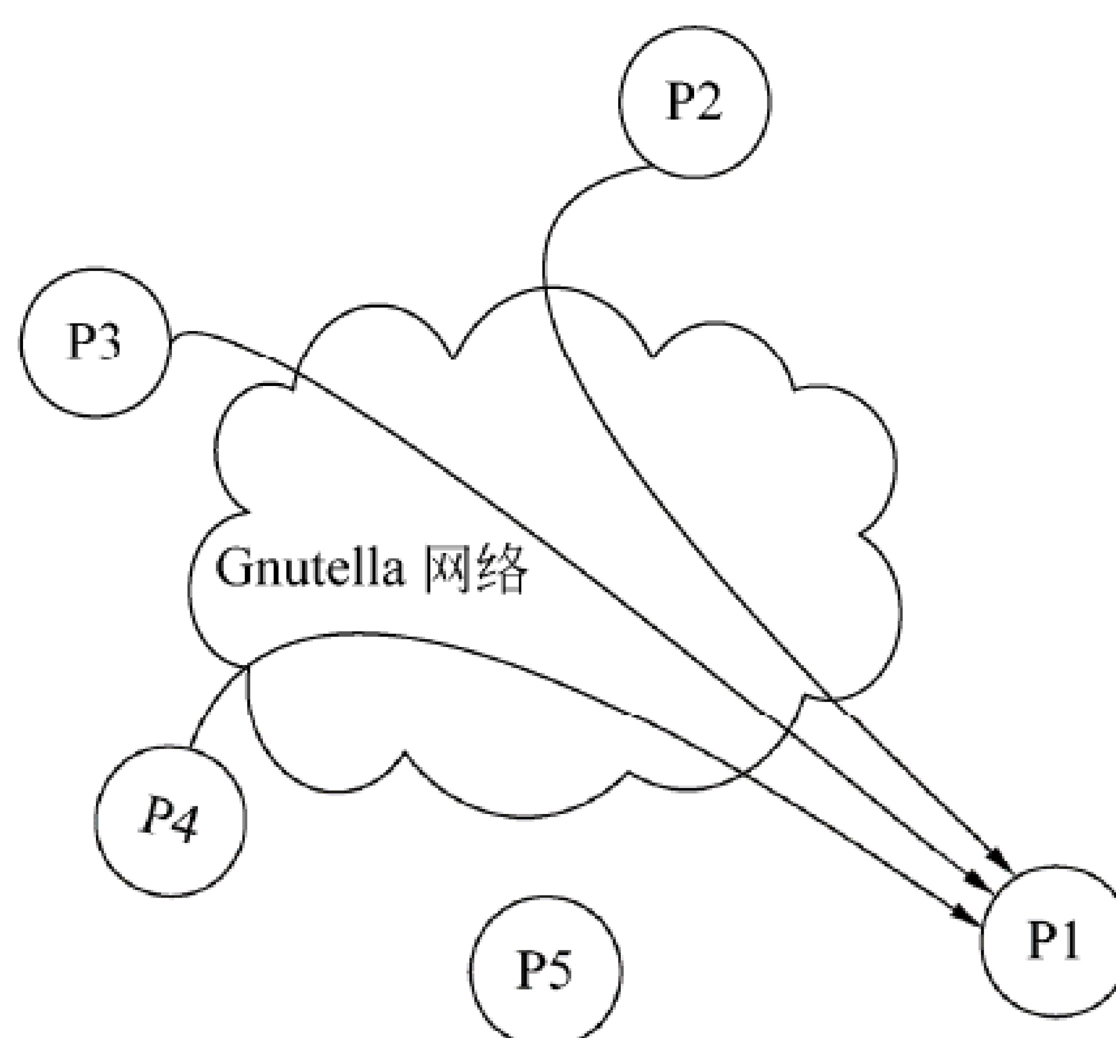


图 4.10　多源的 P2P 流媒体模型

4.5　P2P 的系统安全技术

P2P 系统存在许多安全问题，从系统本身的角度看，由于 P2P 网络的开放性、用户发布消息的匿名性及 Ad Hoc 的连接方式，使得 P2P 系统存在许多安全隐患。从多用户的角度看，由于在 P2P 网络中结点的硬盘需要被其他结点访问，结点用户安全意识的缺乏及 P2P 协议和 PC 操作系统的安全漏洞使得结点很容易受到黑客攻击。P2P 中的安全问题直接决定了 P2P 能否被大规模进行商用，这些就要求必须在 P2P 系统中设计优良的安全机制来解决这些问题。

注意： Ad Hoc 网络是一个没有有线基础设施支持的移动网络。在 Ad Hoc 网络中，所有的结点都是由移动主机构成的。该类型的网络最初是应用于军事领域，是为了在战场环境下分组无线网络数据的通信。网络拓扑结构的动态性是 Ad Hoc 网络的重要特点。Ad Hoc 网络通信的核心问题在网络通信效率和结点能量消耗之间的合理平衡。

4.5.1　不安全的问题

在 P2P 系统中，不安全的问题有很多，由 P2P 系统本身的性质引起的，也有主动恶意攻击造成的。例如，在采用复制技术的 P2P 系统中，P2P 系统一般都提供了复制特性，恶意用户可以利用这个特点，无限制地产生恶意文件在网上传播。Gnutella 系统中的恶意用户就可以产生 VBSGnutella 蠕虫病毒，并通过自我复制，任播到系统中的多个结点中用于共享，破坏系统。

还有一类问题就是结点被动受到攻击，这一般会造成两种后果。

- 用户个人信息被剽窃，文件被病毒感染，其他从该结点下载文件的用户也会被感染。

- 用户站点被黑客控制成为分布式拒绝服务的发起者。在商业中，使用这种结构共享关键数据可能导致严重的安全问题。

以上只是对个别安全问题进行大致的说明，本书有专门的章节对 P2P 的安全问题进行细致的讲解。

4.5.2　P2P 平台的匿名性

对于 P2P 存在的诸多安全问题，已经有多种机制来解决，大部分的 P2P 系统中都采取了身份认证、数字签名、加密算法、防火墙等技术来保障用户和系统的安全。

在 P2P 技术中，与安全有紧密关系的是 P2P 平台提供的匿名性。下面就重点讲一下 P2P 的匿名技术。

为了进行自由的交互，避免引起不必要的法律纠纷，所以，P2P 系统允许匿名方式的信息交换。匿名性可分为以下 3 种类型。

- 发布者匿名：发布者匿名地发布服务或资源。
- 请求者匿名：请求方匿名地请求服务或资源。
- 服务提供者匿名：P2P 网络中的资源拥有者匿名地提供资源或服务。

提供匿名性的基本方法有 6 种。

- 组播：采用 IP 组播方法进行资源发送可以达到请求者匿名的目的。但是这种方法需要底层网络如以太网的支持。
- 地址欺骗：使用面向无连接协议如 UDP，可以更改地址，达到地址欺骗的目的。
- 身份欺骗：网络中的文件可能拥有多个备份，所以应答者往往并不是文件的提供者，如 FreeNet 就使用了这种方法提供匿名性。
- 隐藏路径：Peer 之间并不是进行直接的消息通信，而是通过若干的中间结点，这样可以达到发送者匿名的要求。
- 别名：每一个 Peer 在网络中均有一个别名，Peer 之间通过别名进行消息传递，这种方法需要一个 Proxy Server 保存所有的 Peer 实体与别名的对应关系。
- 强制存放：一个文件存放的位置是由文件本身确定的，发布者无法选择，从而实现发布者匿名，如基于 Chord 的系统。

上述方法可以组合应用，例如，FreeNet 采用纯 P2P 网络中的非结构化文档路由模型，同时在发布文件和查找文件的过程中，沿搜索路径复制文件。通过隐藏路径和身份欺骗的方法达到服务提供者匿名的目的，通过隐藏路径的方法形成请求者匿名，同时由于文件的发布是一个根据文件名强制存放的过程，所以也完成了发布者匿名的功能。

4.5.3　难以避免的安全隐患

P2P 系统虽然有其自身很多独特的优点，也在很多方面得到应用，但就安全这一点来说，它也存在着某些难以避免的安全隐患。

- 拒绝服务攻击：P2P 应用会消耗网络带宽，占用磁盘空间，使得系统性能降低甚至完全瘫痪。如果被大量恶意地使用，就会出现正常服务请求得不到服务的现象。处理这种攻击的难点在于如何将恶意结点和真正负载过重的结点区分开。从根本

上说，限制网络服务请求的数量是解决拒绝服务攻击的根本方法。合理选择网络底层的拓扑结构，均衡系统中的负载和资源，加入流量控制策略，这三种措施综合使用可使恶意结点根本无法占用过多的资源，降低其对系统的影响。

- 恶意软件分发：P2P 无中央服务器控制，允许匿名分发资源，对软件的分发缺乏控制，如果 P2P 系统被恶意使用，会给系统用户带来安全隐患。通常的名誉评估系统可以用来跟踪和处理 Peer 的状态，并根据得到的信息选择资源的提供者。但是这种方法比较消耗网络资源。
- 信息泄露和篡改：例如，有些 P2P 系统在路由表中保存了一批网络地址，这些信息可能会被泄露和篡改；有些 P2P 系统例如 Napster、Gnutella 允许直接访问用户硬盘，恶意用户可利用这个特性察看和篡改磁盘信息。这些都需要有相应的策略加以解决，例如，设计安全路由策略，可解决路由表信息泄露问题。

P2P 作为一种新兴的技术，从其萌芽到发展到实际应用的过程中，都解决了很多关键问题，也积累了许多宝贵的技术经验，以上所说的只是对 P2P 整个技术体系中所涉及的基础技术要点的罗列。P2P 技术在发展过程中还有引发了很多高级的、深层次技术的突破和变革，也带动了相关领域和行业的发展，在当前的互联网发展和应用中展现出了巨大的价值。当然 P2P 也暴露出了不少问题，还需要我们不断地努力，不断地突破，以寻求 P2P 技术更大的发展。

4.6　P2P 应用及其典型应用系统

P2P 已成为当今科学研究和产品开发的热点，在网络电视、文件共享、分布式计算、网络安全、在线交流甚至是企业计算与电子商务等应用领域上，P2P 都显露出了很强的技术优势。基于 P2P 的系统和应用软件也遍布网络，像迅雷、eMule、BT、Skype 等，大部分网民几乎每天都会用到，可以说，P2P 技术已经与人们的日常生活紧密联系起来了。

4.6.1　P2P 的应用分类

基于 P2P 的应用可以按照不同的方式、不同的性质分为不同的类别。在 P2P 的技术领域，可以将 P2P 的应用进行如下分类：

- 文件共享（File sharing）；
- 语音通信（VoIP）；
- 流媒体技术（Streaming media）；
- 即时通信和在线聊天技术（Instant messaging and online chat）；
- 软件的发布与分发（Software publication and distribution）；
- 匿名 Web 协议的创建（Creation of anonymous web portals）；
- 流媒体的发布与分发 Media publication and distribution（radio，video）。

本文按另一种分类方法来讲解 P2P 的应用，总的分类包括 4 大方面，分别为：

- 内容共享应用（content-sharing）；
- 分布式计算应用（distributed computing）；

- ❑ 通信与协作应用（communication and collaboration）；
- ❑ P2P 系统平台（Platform）的开发应用。

P2P 应用的基本分类图如图 4.11 所示，基本上包括了大部分的 P2P 应用。

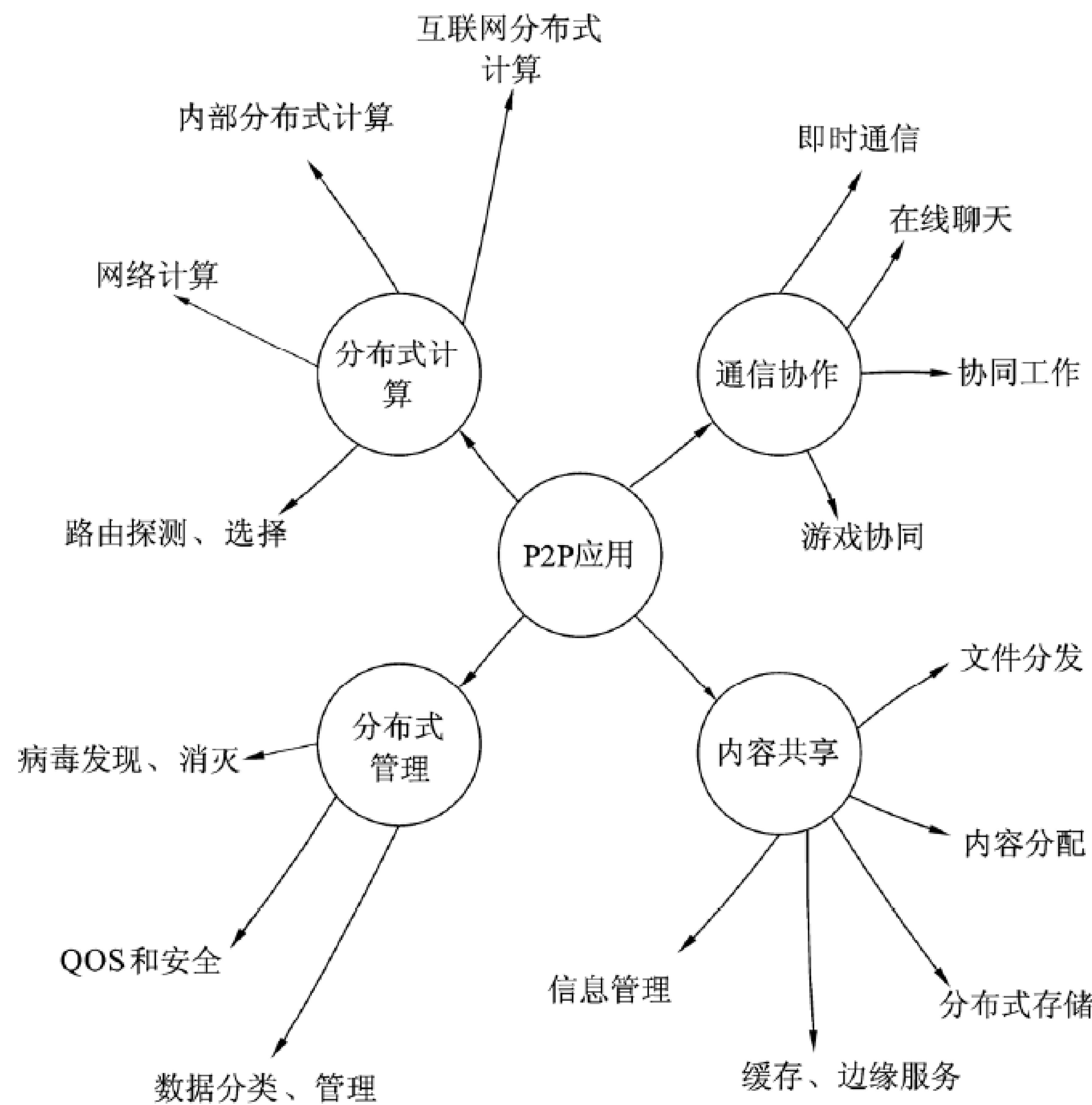


图 4.11　P2P 应用的基本分类图

不管以哪种方式分类，P2P 技术典型应用的整体情况不会改变，下面会针对一些常用的、重点的应用技术进行讲解。

4.6.2　P2P 的应用平台

基于 P2P 的应用非常广泛，其应用平台也是应有尽有，下面就是对 P2P 涉及的应用平台进行说明，如表 4.1 所示。

表 4.1　P2P的应用领域及代表性的软件平台

网络及协议	应 用 领 域	应用软件/平台
ANtsP2P	文件共享，软件分发，流媒体分发	ANtsP2P
Ares	文件共享	AlterGalaxy,AresGalaxy,WarezP2P,KCeasy
BT	文件共享，软件分发，流媒体发布	ABC,AllPeers,Vuze(formerlyAzureus),BitComet,BitLord,Bit Tornado,BitTorrent,Burst!,Deluge,FlashGet,G3Torrent,Halite,KTorrent,LimeWire,MLDonkey,Opera,QTorrent,rTorrent,Shareaza,TorrentFlux,Transmission,Tribler,µTorrent,Xunlei

续表

网络及协议	应 用 领 域	应用软件/平台
DirectConnect	在线聊天，文件共享	DC++,NeoModusDirectConnect,SababaDC,BCDC++,RevConnect,fulDC,LDC++,CzDC,McDC++,DCDM++,DDC++,iDC++,IceDC++,Zion++,R2++,rmDC++,LinuxDC++,LanDC++,ApexDC++,StrongDC++
Domain Name System	网络信息交互	See Comparison of DNS server software
eDonkey	文件共享	aMule,eDonkey2000(discontinued),eMule,eMulePlus,FlashGet,Jubster,lMule,MLDonkey,Morpheus,Pruna,Shareaza,xMule
FastTrack	文件共享	AlterGalaxy,giFT,Grokster,iMesh(prev6)(anditsvariantsstrippedofadwareincludingiMeshLight),Kazaa(anditsvariantsstrippedofadwaresuchasKazaaLite),KCeasy,Mammoth,MLDonkey,Poisoned
Freenet	分布式存储	Entropy(on its own network),FreeNet
GNUnet	文件共享，在线聊天	GNU net,(GNU net-gtk)
Gnutella	文件共享	AlterGalaxy,Acquisition,BearShare,Cabos,FilesWire,FrostWire,Gnucleus,Grokster,gtk-gnutella,KiwiAlpha,LimeWire,MLDonkey,Morpheus,MP3Rocket,Poisoned,Shareaza,Swapper,XoloX
Gnutella2(G2)	文件共享	Adagio,Gnucleus,KiwiAlpha,MLDonkey,Morpheus,Shareaza,Sharelin,TrustyFiles
JXTA	P2P开发平台	CollanosWorkplace(Teamworksoftware),Sixearch
KadNetwork	文件共享	aMule,eMule,MLDonkey
Napster	文件共享	Napigator,Napster
OpenNap	文件共享	WinMX,Utatane,XNap,Napster
P2PTV	流媒体播放，文件共享	TVUPlayer,Joost,CoolStreaming,Cybersky-TV,TVants,PPLive,Live Station
PDTP	流媒体播放，文件共享	PDTP
Peercasting	多播流	PeerCast,IceShare,FreeCast,Rawflow
Usenet	分布式讨论组	See list of news clients
WWIVnet	分布式网络	SeeWWIV
WPNP	文件共享	WinMX

4.7　P2P 基于内容共享的应用

内容存储和共享是 P2P 技术应用最为成功的领域之一。作为 P2P 网络系统，“共享”永远是其追求的目标，在 P2P 网络中，参与共享系统中的计算机之间可以直接交换和共享自身的资源。这里的资源包括存储的文件、CPU、数据以及存储空间等，根据共享的内容不同，P2P 中的共享可以分为以下几类。

4.7.1　数据文件共享

在 P2P 应用系统中，以数据文件的共享最为常见，这些文件包括音频、视频、图像等

多种文件形式。文件共享有两种基本方式：基于目录服务的文件共享和对等式文件共享。前者典型的例子为 Napster，后者的典型例子是 Gnutella、FreeNet。

1．Napster

1999 年，18 岁的肖恩范宁（Shawn Fanning）开发出了一种用于共享 MP3 文件的软件 Napster。Napster 还是一种不完全的 P2P 系统。在 Napster 中，需要一组中央服务器来保存系统中所有共享文件的索引。每一个用户都与其中的一个服务器相连，并且通过这个服务器传送查询请求。索取文件时，用户通过所需文件的文件名向服务器发出请求，服务器收到请求后，将与其他的服务器共同合作，通过索引完成查询工作，并将找到的保存有所需文件的用户 IP 地址返回给请求发起用户。然后由发起请求的用户与目的用户直接通信，完成文件的交换。如图 4.12 所示，是 Napster 的工作示意图。

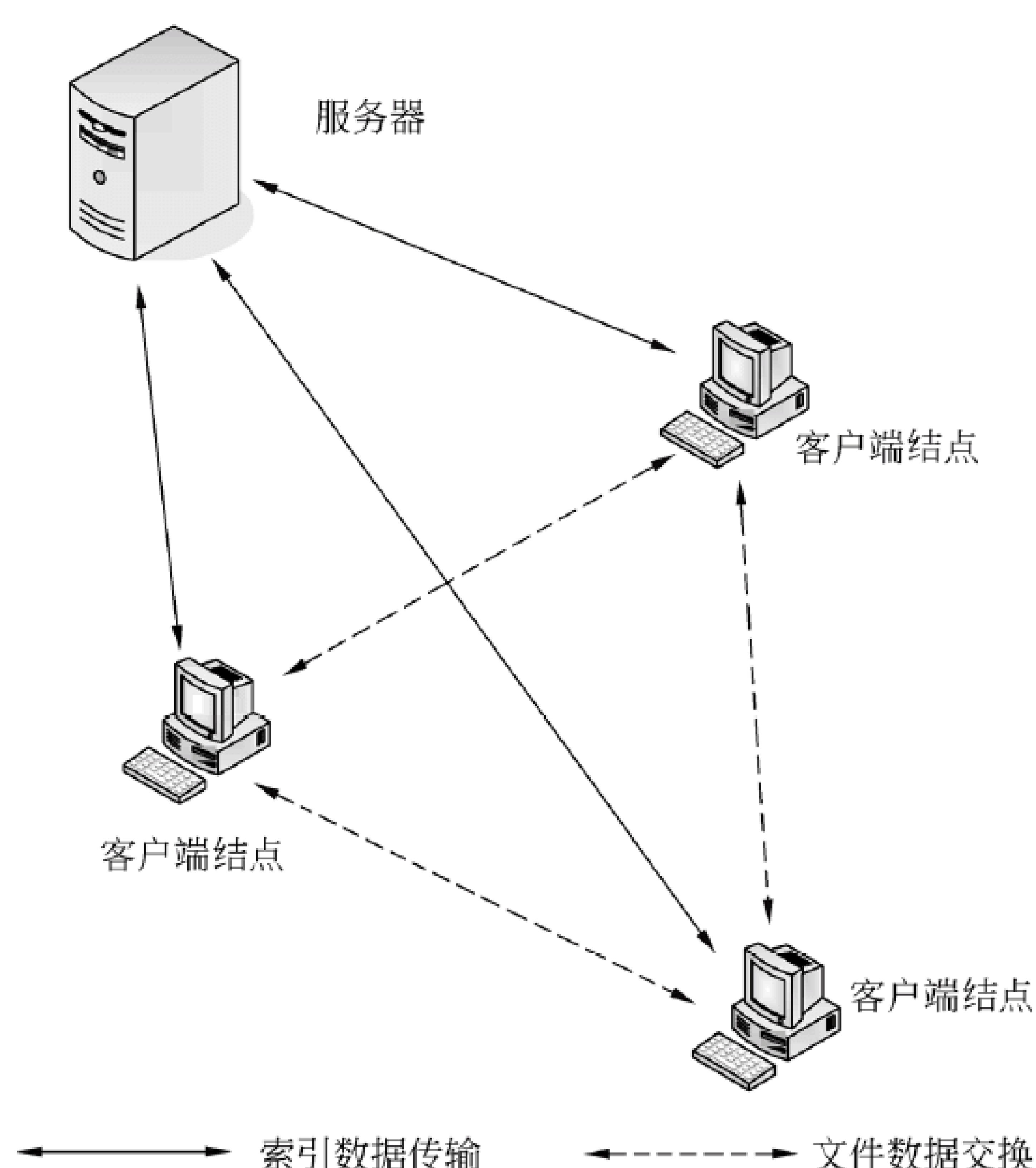


图 4.12　Napster 工作示意图

2．Gnutella

Gnutella 中没有集中的服务器，系统中的每一个用户都与自己的一组邻居用户结点通过端对端连接，构成一个逻辑覆盖的网络［3，5］。查询文件时，用户向自己所有的邻居结点发送查询数据包。每一个收到查询数据包的用户将检查在自己本地存储的文件是否满足查询要求。如果满足的话，该用户将发送一个查询响应数据包给查询的初始发起者，然后两个用户之间直接交换文件。不管满足与否，该用户都继续将查询数据包向自己的邻居结点转发，依次类推，查询消息像洪水（Flood）一样在网络中流动。洪水流动的范围是由查询消息的 TTL 控制的，随着查询消息的向前传输，TTL 值将会减少，TTL 减少为 0 时，查询消息将停止传送。

3．FreeNet

FreeNet 文件共享系统的主要设计目标是提供一种匿名的方法，对信息进行存储和索取。系统中的用户在发布文件和发起索要文件的请求时，其他用户都不能准确地知道文件或请求的初始发起者。FreeNet 系统中的每个结点都有一个 ID 序列号，每一个文件也有一个通过 SHA-1 哈希函数映射得来的 ID 序列号。文件查询时，用户将查询请求，转发给拥有与文件 ID 最相近的 ID 的那个用户结点，直到找到所需的文件。

4.7.2　CPU 共享

P2P 系统中 CPU 共享其实就是计算能力的共享，在实际应用中，以共享计算能力为目标的 P2P 系统也不少，如 SETI@home。

SETI 是英文 Search for Extraterrestrial Intelligence（搜寻外星智能）的缩写。该项目试图通过分析 Arecibo 射电望远镜采集的无线电信号，搜寻能够证实外星智能生物存在的证据。该项目由美国加州大学伯克利分校的空间科学实验室主办。

SETI@home 程序在用户的个人计算机上，通常在屏幕保护模式下或以后台模式运行。它利用的是多余的处理器资源，不影响用户正常使用计算机。SETI@home 项目自 1999 年 5 月 17 日开始正式运行。至 2004 年 5 月，已经积累了近 200 万年的 CPU 运行时间，进行了近 5×1021 次浮点运算，处理了超过 13 亿个数据单元。

目前共有 226 国家和地区、超过 500 万的个人和团体参加了这项浩大工程，可以到 BOINC 项目的主页上查看当前的详细信息，也可自由地参与到这个项目中去，主页地址为 http://boinc.berkeley.edu/。

4.7.3　存储空间共享

存储共享是利用整个网络中闲散的内存和磁盘空间，将大型的计算工作分散到多台计算机上共同完成，这样可以有效地增加数据的可靠性和传输速度。

以存储空间为共享资源的系统有 OceanStore，它是以 Tapestry 为路由和查找基础设施的 P2P 平台，是一个适合于全球数据存储的 P2P 应用系统。如图 4.13 所示，是一个基于全球分布的 OceanStore 系统模型。

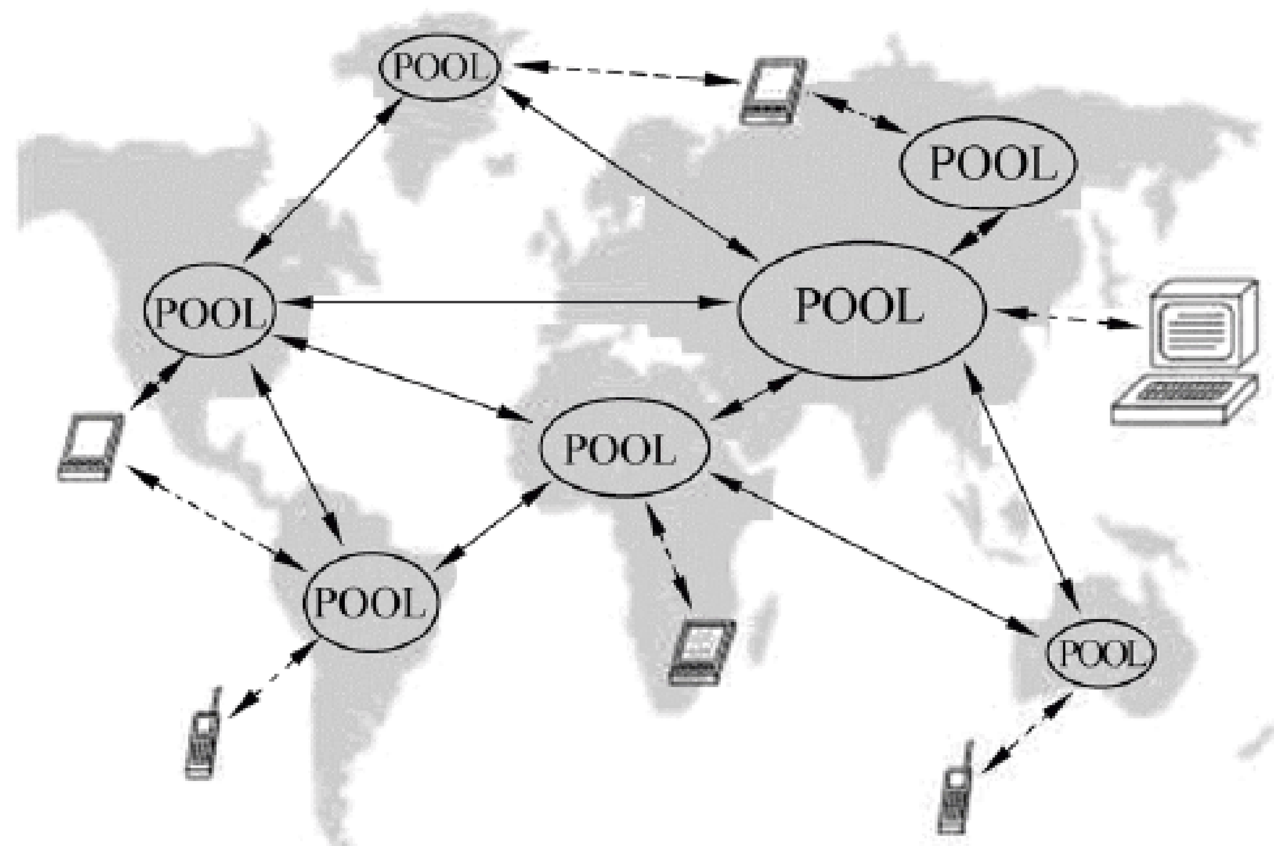


图 4.13　OceanStore 系统模型

在 OceanStore 系统中，可以实现能提供全球分布式的、持续稳定的数据存储。用户计算机上的每一个文件都被分配一个全球唯一标识（GUID），然后由软件将文件分割成许多微小的片断进行加密，加密的文件被分解成为互相重叠的片断存储在全球 ISP 拥有的众多 Web 服务器中。即使一些本地的结点损坏，也可以通过一组片断恢复原始的文件。

OceanStore 对文件片断进行加密，使得只有在用户端才能看到文件的明文，服务器上看到的只是加了密的密文，有效地保证了文件的安全性。

4.8　内容下载与分发

在 P2P 系统中，还有一类应用就是内容的分发与交换，它也是内容共享的一种，只是这种共享更强调相互之间的数据交换上。

4.8.1　文件下载

基于 P2P 的文件下载，可以说是目前 P2P 技术中应用最普遍，使用最广泛的技术。P2P 的下载原理其实很简单，它不同于通常所说的 Web 下载，在 P2P 系统中，每个参与的 Peer 都把自己的文件共享出来。当其他的 Peer 搜索到这个文件时，两个 Peer 之间就直接建立连接，进行文件传输，而不需要任何第三方的参与。这样，当所有的 Peer 都参与进来时，就可以实现下载的人越多，下载速度反而越快的目的，突破了普通 Web 下载时服务器负载过重的瓶颈，充分地利用了带宽。

利用 P2P 技术来进行文件下载的系统有很多，常用的有 Napster、Gnutella、eDokney、eMule 和 BT 等。

要问一百个网友目前中国最流行的文件下载方式，恐怕 99 个都会回答是 BT。BT 是一种依赖 P2P 方式将文件在大量互联网用户之间进行共享与传输的协议，对应的客户端软件有 BT、（简称 BC）和 BitSpirit（简称 BC）等。

由于其实现简单、使用方便，在中国用户之间被广泛使用。BT 中的结点在共享一个文件时，首先将文件分片并将文件和分片信息保存在一个流（Torrent）类型文件中，这种结点被形象地称作“种子”结点。其他用户在下载该文件时根据 Torrent 文件的信息，将文件的部分分片下载下来，然后在其他下载该文件的结点之间共享自己已经下载的分片，互通有无，从而实现文件的快速分发。如图 4.14 所示为 BT 的文件下载过程示意图。

由于每个结点在下载文件的同时也在为其他用户上传该文件的分片，所以整体来看，不会随着用户数的增加而降低下载速度，反而下载的人越多，速度越快。

注意： *后文会有对 BT 的专门讲解，在本书的实践开发部分也会带领读者一起开发一个 BT 下载的客户端。*

还有另一类不同于 BT 形式的下载就是 KaZaA 下载。KaZaA 也是一种基于 P2P 技术的文件共享协议，在 KaZaA 协议中，结点是无结构连接的。但是存在一种“超级结点”。这种“超级结点”其实是来源于各个普通的客户端结点，但它们一般具有计算能力强、接入带宽大、在线时间稳定等特点。在 KaZaA 协议中，每个结点之间并不是完全对等的，超

级结点承担着部分服务器的任务，如管理部分普通结点、负责搜索消息的转发等。

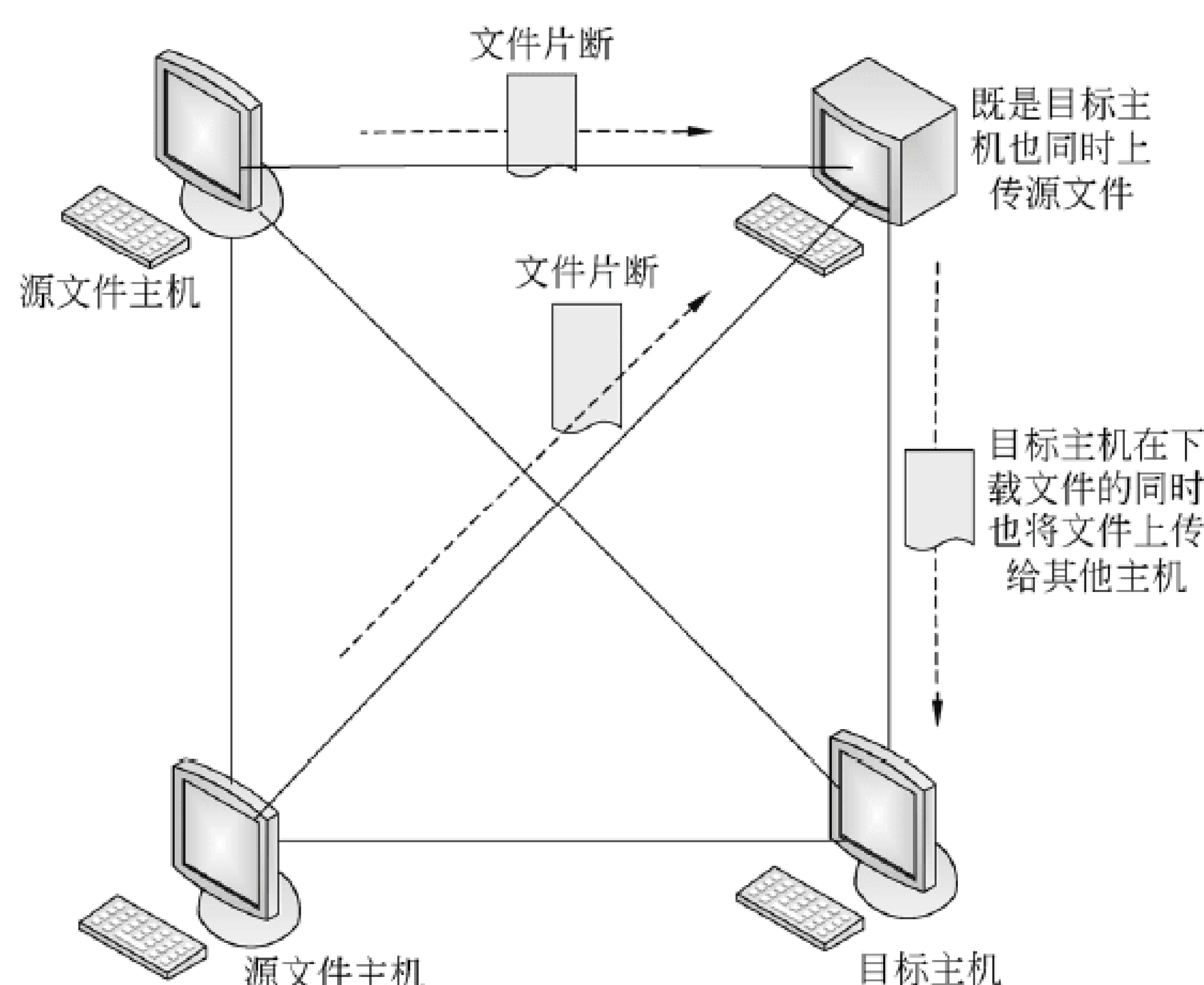


图 4.14　BT 下载过程示意图

使用 KaZaA 时，一旦有结点上线，就会寻找一个超级结点挂靠，并和原先挂靠在该超级结点下的其他普通结点随机相连，组成一个小的无结构网络。普通结点的共享文件索引汇报给所挂靠的超级结点。因而，KaZaA 网络大体上可以看作是两层的无结构网络，上层是超级结点组成的无结构网络；下层是普通结点组成的多个无结构网络，按所挂靠的超级结点分成多个簇。

当一个普通的结点发起文件搜索请求时，将请求消息发给所挂靠的超级结点，超级结点从自己存储的共享文件索引信息中查找区域内符合条件的文件，同时将搜索请求转发给若干个其他超级结点，由它们返回其区域内搜索结果。如果需要，这个转发过程可以执行多步以获得更大范围内的搜索结果。这样的混合式结构对异构的终端结点“分而治之”，可以充分利用一些能力较强的终端结点来担任“小”服务器的角色，可谓是“人尽其才，物尽其用”。

除了这些无结构的 P2P 文件共享协议之外，几乎所有的 DHT 网络都可以并已经用来实现文件共享的应用，如 Chord、Pastry、KAD、CAN 等应用。

4.8.2　流媒体分发

前面已经说过流媒体的定义，简单的说流媒体就是流式数据，在 P2P 的技术平台上进行流式数据的发布就是流媒体分发技术，基于 P2P 的直播技术、P2P 点播技术都是可以说是 P2P 的流媒体分发技术。

1．流媒体直播

曾经人们以为 P2P 做文件共享最合适，但现在大家发现 P2P 模式是如此适合于流媒体直播，以至于研究热点在很短的时间内迅速转移到 P2P 的流媒体上来。中国最早的 P2P 流

媒体直播软件应该算香港科技大学计算机系研究的 Coolstreaming、AnySee 以及 Gridmedia 等系统。

以 AnySee 为例，它所设计的初衷是期望 AnySee 软件能够使得用户在网上任何时候任何地点都能观看多媒体直播节目。图 4.15 为 AnySee 系统运行部署图。

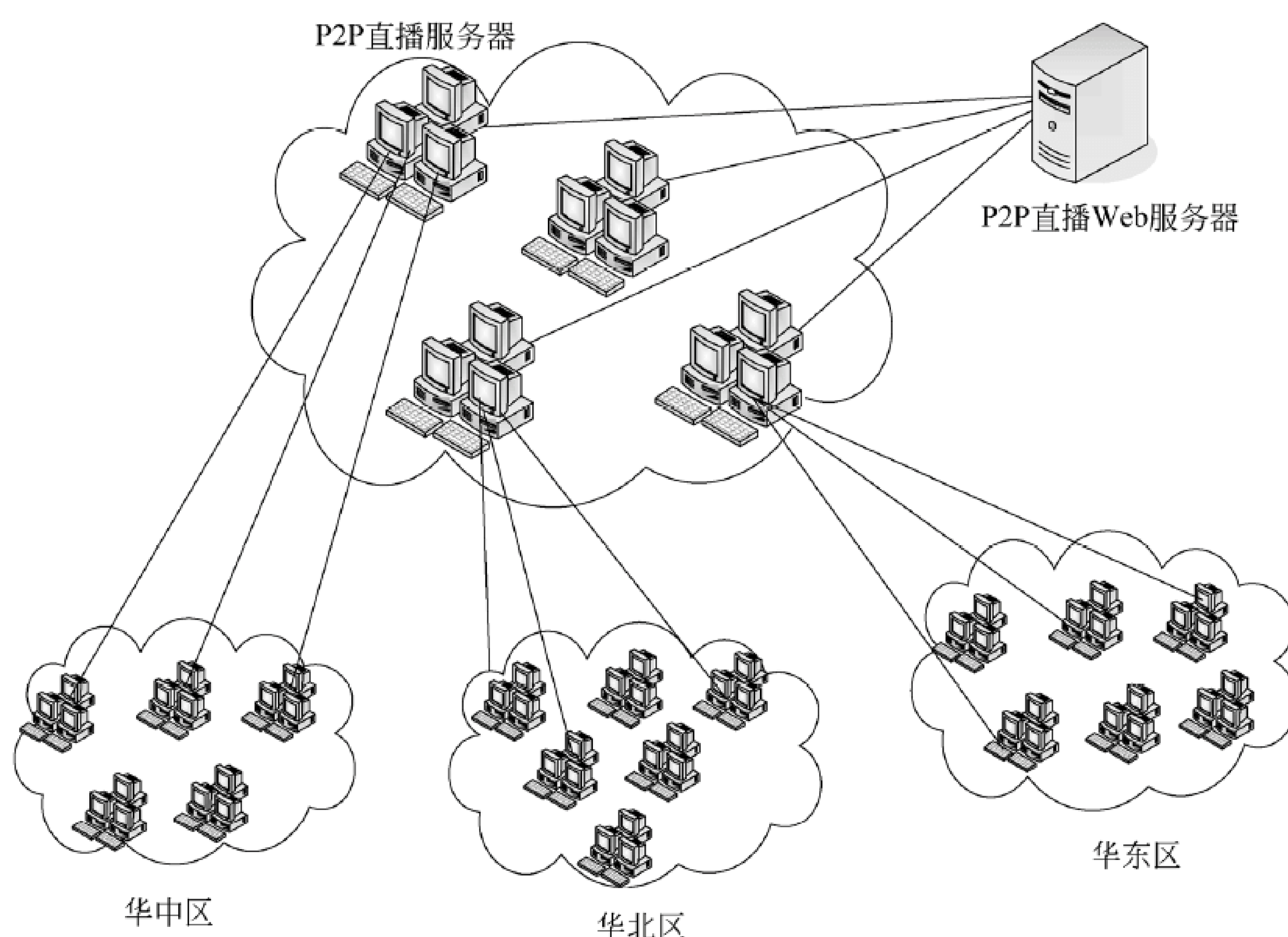


图 4.15　AnySee 系统运行部署图

AnySee 的第一个版本基于树状结构：节目源是一个多播树的根结点，之后的结点被调度为其“儿子”或子树。每个结点向其父结点索要数据，并将数据提供给多个子结点。这样的结构可以使得结点快速加入到网络中，并且可以根据 IP 邻近原则构建起一棵 IP 多播树，使得结点加入位置都是和自己 IP 邻近的结点，从而优化服务质量。

随后，AnySee 推出了第二个版本，在第二版 AnySee 中，结合了原有的树状结构和流行的网状结构，使得“控制数据走树，媒体数据走网”，既能帮助结点快速定位到加入点，又能实现一定程度的负载均衡，并缓解了原有纯树状结构中底层结点和顶层结点之间播放时差较大的问题。

在最新的 AnySee 版本中，已经取消了树的结构，演化成了优化的网状结构（如图 4.15 所示），即每个结点维护一定数量的邻居成员，并从中选出最合适的“伙伴”结点与之交换数据。伙伴的数量既有上限又有下限，在不满足下限时，结点会不断寻找新的合适结点加入伙伴列表；在达到下限时，结点停止主动寻找伙伴的过程，但可以接受其他结点将其加入伙伴列表的请求；在达到上限时，结点不再和新的结点建立伙伴关系。

除了学术界对 P2P 流媒体直播的研究外，中国还涌现了很多成功的 P2P 流媒体直播商业产品，如 PPLive、PPStream、沸点和 TVAnts 等，其中以 PPLive 最为有名。PPLive 目前拥有数百个频道，在 2006 年“超级女声”决赛期间，频道观看人数达到十万人，可以说是把 P2P 发挥到了极限。此外，国外也有不少对 P2P 流媒体直播的研究，如 SplitStream 等。

2．流媒体点播

由于观看直播节目时用户不能选择观看指定片段，所以在人们热烈研究 P2P 流媒体直播时，已有人开始将目光转向 P2P 流媒体点播服务。图 4.16 展示了一个基于 P2P 的点播系统架构图。

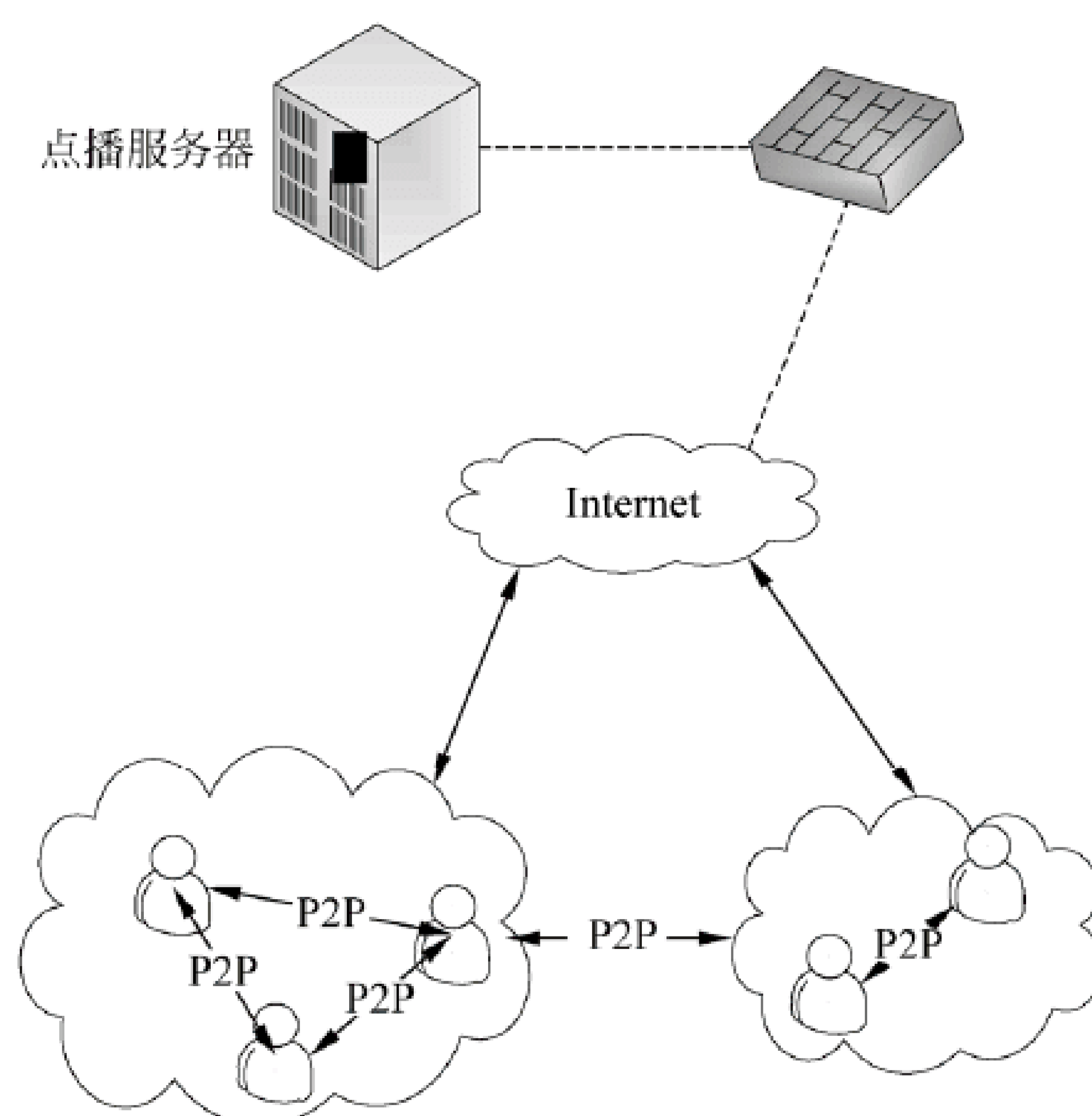


图 4.16　P2P 的点播系统架构图

目前成功推出 P2P 流媒体点播的机构还不多，典型的有 GridCast 系统、PPStream 点播系统。

GridCast 视频点播系统具有支持多人共享点播片段、跟踪（Tracker）服务器用户引导、环状结构内容组织等特点。由于一个点播频道的人数往往不会太多，所以在用户进行视频录放（VCR）操作时（即前后拖动播放点、暂停/继续播放等操作），能否快速将用户定位到观看该点节目的其他用户处就成了 P2P 点播技术的关键。为了实现快速定位，GridCast 中采取了一种同心圆环的媒体内容组织结构。在每一个节目频道里，媒体内容按指数递增的区间进行划分，例如一个一个半小时的电影节目，可划分成[0，5]、(5，15]、(15，35]、(35，75]和(75，END=90]几段，其单位为分钟。每个结点记录几个正在观看各个段之间内容的结点。这样，在和 AnySee 类似的网状结构中，可以定期交换这种分段记录，从而，在某个用户拖动观看点时，可以快速定位到相应段的记录结点处，并从这些结点当时所观看的区间内得到大量备用记录以请求该区间媒体数据。此外，GridCast 还根据用户习惯对数据调度策略进行优化。

4.9　分布式计算

分布式计算是一门计算机科学，它研究如何把一个需要非常巨大的计算能力才能解决的问题分成许多小的部分，然后把这些部分分配给许多计算机进行处理，最后把这些计算结果综合起来得到最终的结果。P2P 技术另一个重要的应用就是分布式计算，通过 P2P 技术，

可以利用整个网络上的计算机的闲置中央处理器、内存以及磁盘空间等，进行大规模的运算。

4.9.1　什么是分布式计算

P2P 的分布式计算通过众多计算机来完成超级计算机的功能，一直是科学家梦寐以求的事情。采用 P2P 技术的对等计算，正是把网络中的众多计算机暂时不用的计算能力连接起来，使用积累的能力执行超级计算机的任务。任何需要大量数据处理的行业都可从这种分布式计算中获利。

在实际应用中，有很多需要非常巨大的计算能力才能解决的问题。这类问题一般是跨学科的、极富挑战性的、人类亟待解决的科研课题。其中较为著名的有：

- ❑ 解决较为复杂的数学问题，例如 GIMPS（寻找最大的梅森素数）。
- ❑ 研究寻找最为安全的密码系统，例如 RC-72（密码破解）。
- ❑ 生物病理研究，例如 Folding@home（研究蛋白质折叠，误解，聚合及由此引起的相关疾病）。
- ❑ 各种各样疾病的药物研究，例如 United Devices（寻找对抗癌症的有效的药物）。
- ❑ 信号处理，例如 SETI@Home（在家寻找地外文明）。

从这些实际的例子中可以看出，这些项目都很庞大，需要惊人的计算量，仅仅由单个的电脑或是个人在一个能让人接受的时间内计算完成是绝不可能的。在过去，这些问题都应该由超级计算机来解决。但是超级计算机的造价和维护非常昂贵，这不是一个普通的科研组织所能承受的。通过 P2P 技术构建的分布式计算架构，可以实现廉价的、高效的、维护方便超级计算能力。

4.9.2　P2P 分布式计算的优点

基于 P2P 的分布式计算，它所参与计算的并不只是一台计算机，而是一个计算机网络。当所有的计算机参与进来的时候，这种“蚂蚁搬山”的方式将具有很强的数据处理能力。这种分布式计算相比其他算法具有以下优点：

- ❑ 稀有资源可以共享。
- ❑ 通过分布式计算可以在多台计算机上平衡计算负载。
- ❑ 可以把程序放在最适合运行它的计算机上。

其中，共享稀有资源和平衡负载是计算机分布式计算的核心思想之一。

4.9.3　基于 P2P 的分布式计算需要解决的问题

基于 P2P 的广域网分布式并行计算，有两个问题需要解决。

- ❑ 协同机制。客户端之间如何互相协作、如何分解问题、如何解决通信导致的延迟、如何实现“热插拔”的问题。
- ❑ 信用机制。WebService 租用和 ASP（Application Service Provider）未能变成主流市场，很大程度上是因为信用机制不够健全。信用有两个层面，一个是信用观念，

一个是信用技术，前者取决于后者。

除了一些科研项目外，目前尚不存在影响力广泛的广域网分布式并行计算应用。P2P 技术的应用，仍停留在较低层次的文件共享上。最有可能让 P2P 分布式并行计算得以实现的是操作系统厂商。《黑客帝国》的英文原名是 Matrix，即矩阵，该片描写的，其实就是一个 P2P 并行运算的场景。片末，人类和计算机达成妥协，或许也预示了 P2P 并行运算的光辉未来。

P2P 分布式并行计算，离我们并不遥远。它得以广泛应用之日，也将是软硬件体系架构大变革之时。那一天，CPU 将和我们的大脑一起被解放。

4.10　通信与协作

P2P 协同应用的目标是允许用户间在应用层上协同工作。这种应用的范围很广，包括即时通信、聊天、在线游戏等。比较典型的有 Groove 系统，OICQ，ICQ 聊天软件等，它们都是 P2P 技术在通信与协作方面的典型应用。

4.10.1　P2P 即时通信

Ganrter 研究中心的关于 2006 年的十大战略性技术的列表清单之一就有即时消息技术。欧洲的 Skype 即时通信软件，成功利用了网络中的所有可用资源，使得 Skype 网络中的通话完成率及音质远远超出旧的普通电话系统。功能提高的同时，还无须成本高昂的中央资源。

注意：在中国大陆，Skype 与 TOM 集团旗下北京讯能网络有限公司 TOM 在线合作，所推出的 Skype 又称为 TOM & Skype。在中国台湾是与网络家庭（PChome Online）合作，推出的 Skype 称为 PChome & Skype。在香港，Skype 与和记环球电讯合作，推出的 Skype 称为 HGC-Skype。在日本则与 livedoor（活力门）合作。可以说，Skype 在全球范围内均有相当广泛的应用。

腾讯 QQ 历年来战果辉煌几乎无法动摇，即使第 2 名的 MSN Messneger 也与其相差甚远。想必用过互联网的人几乎都用过 QQ，这里就不再多说了，如图 4.17 所示，就是 2008 版的 QQ 系统运行界面图。

除了以上所说的 Skype 和 QQ 外，其他典型的基于 P2P 的即时通信系统有 ICQ、OICQ、AIM、Yahoo Messenger、Crowds 和 Jabber 等。

实时通信技术是网络中重要的通信技术，从某种意义上说，实时通信应用将超过文件共享应用，成为 P2P 网络技术的第一大应用。P2P 实时通信软件不仅可以随时知道对方是否在线，而且交流双方的通信完全是点对点进行，不依赖服务器的性能和网络带宽，结点之间直接进行数据通信。尽管目前的即时通信技术一般都具有中心服务器，但中心服务器仅是用来控制用户的认证信息，帮助完成结点之间的初始连接。例如，Jabber 就是一个开放源码的实时通信平台，它提出了一个采用 XML 表示的、并且能在不兼容的各种实时通信平台之间进行消息交换的协议。

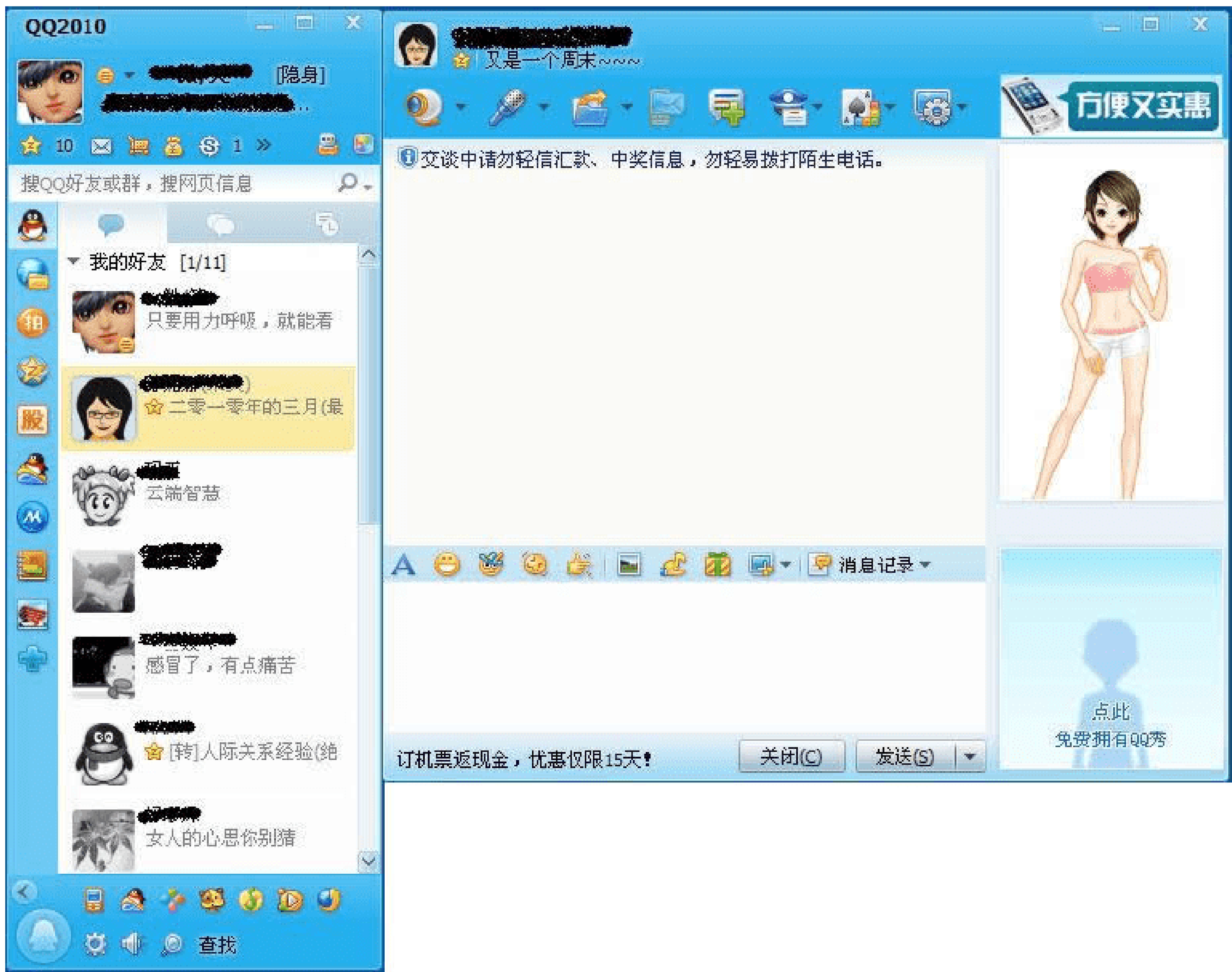


图 4.17　QQ 系统运行界面图

4.10.2　P2P 协同工作环境

协同工作是指多用户之间利用网络中的协同计算平台互相协同来共同完成计算任务，共享信息资源等。通过采用 P2P 技术，个人和组织可以随时采用多种方式建立在线、非在线的协同应用环境。协同应用一般包括实时通信、聊天室、文件共享、语音通信等基本功能，除了这些基本功能，用户之间还可以共享白板、协同写作、视频会议等。Groove 就是基于 P2P 的协同软件平台，已经被微软公司收购。

Groove 系统主要针对 Internet、Intranet 的用户，同时也能为移动设备如 PDA 和移动电话上的用户提供服务。它的设计目标是能够使用户间直接通信而不需要依赖于服务器。在 Groove 体系结构中，Groove 层介于应用逻辑层和命令执行层之间，命令消息在 Groove 层被转化成 XML 对象，然后被存储和排序，这样可以使连接中断时，消息被重新发出。Groove 中的用户是在一个称之为共享空间（Shared Space）的实时环境中进行直接通信的，系统中的用户需要身份认证，数据在磁盘上和通信线路上都要被加密。如图 4.18 所示，就是 Groove 系统在 Windows 平台下的运行情况。

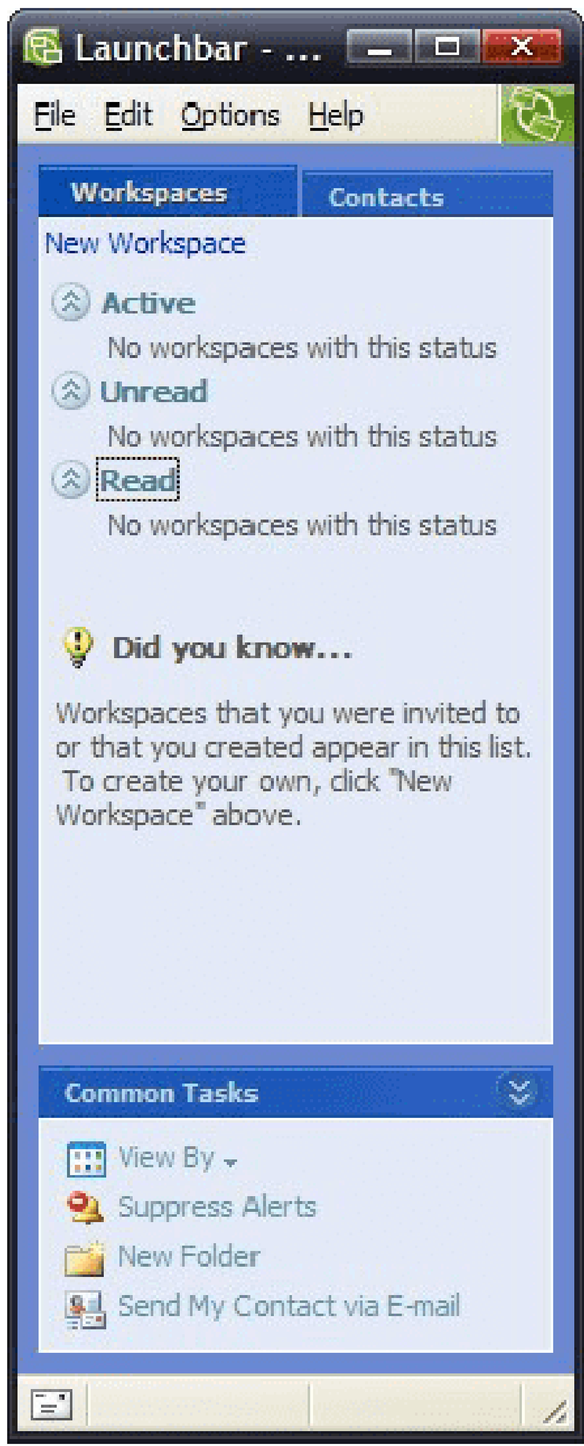


图 4.18　Groove 系统在 Windows 平台下的运行情况

另外，协同有时候还包括工程人员的协作开发软件。例如，JBuilder2006 Java 集成开发环境就增加了 P2P 协同开发的属性。

采用 P2P 技术使协同工作不再需要中心服务器，只要拥有网络，双方存在信息沟通的要求，参与协同工作的计算机就可以点对点建立连接。通过适当的 P2P 软件就能建立协同应用环境。从而避免了中央服务器产生的网络和处理延迟及性能瓶颈。

4.11　P2P 其他的应用

P2P 除了前面介绍的这些技术外，还包括语音通信、组播等，下面进行详细的介绍。

4.11.1　IP 层语音通信

IP 层语音通信（VoIP）是一种全新的网络电话通信业务，它和传统的 PSTN 电话业务相比有着扩展性好、部署方便、价格低廉等明显的优点。在全球范围内的 VoIP 应用中，由于通信各方可能处于不同的网络状况下，所以采取少数几个服务器来进行话音包中转不仅存在压力过大的问题，还可能无法为指定通信双方提供满意的通话质量保证。所以采取 P2P 技术动态自适应地根据通信双方网络进行链路控制与消息转发是可行的解决方案。

刚才所说的 Skype 除了是一个即时通信软件外，也是一款典型的 P2P VoIP 软件。Skype 由于能够提供清晰的语音质量和免费的服务，使用起来又方便快捷，所以吸引了全球数千万的用户，每天在线用户达 500 万人，并且注册用户的数量每天增加 15 万多人。

Skype 由 KaZaA 开发人员所研发，采取类似 KaZaA 的拓扑结构，用 P2P 的技术与其他用户连接，并转发相应的语音通信包。通过 Skype 可以进行高清晰语音聊天，联机双方网络顺畅时，音质可能超过普通电话。如图 4.19 所示为 Skype 的系统结构原理图。

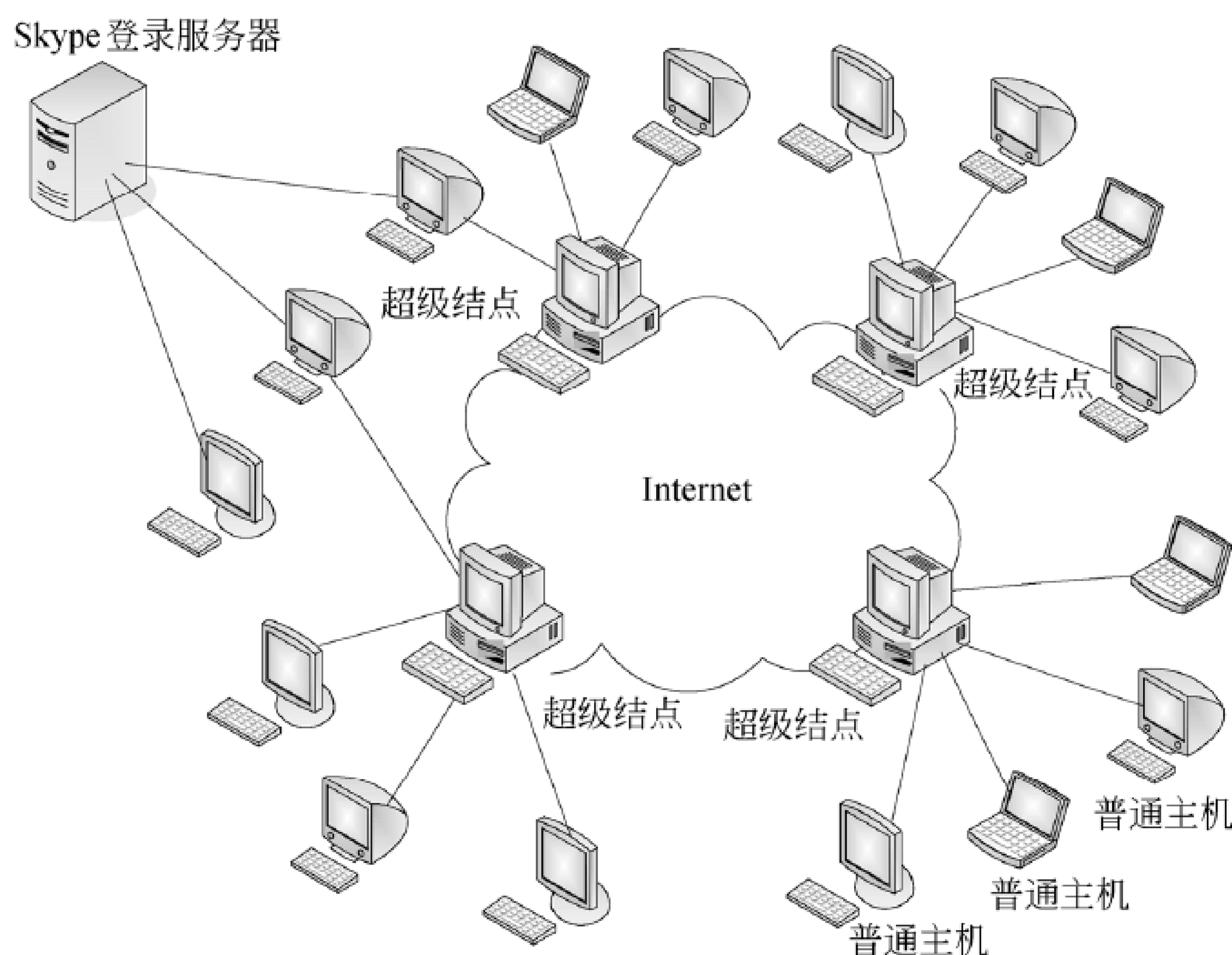


图 4.19　Skype 系统结构原理图

Skype 软件在工作的时候，会在计算机上开启一个网络联机端口来监听其他 Skype 用户的联机呼叫；当其他计算机能顺利联机到这部计算机时，Skype 称呼该用户为 Super node、Super Node 在该 P2P 环境中的角色，即为提供其他无法被联机的用户之间的中继站，借用诸多 Super Nodes 的些许网络带宽，协助其他的 Skype 使用者之间能够顺利的互相联系。这种行为，在 P2P 环境中，这算是相当常见的手法，也是点对点联机的精髓之一。Skype 是第一个将此种做法运用到网络语音通话与即时消息应用层面上。

4.11.2　P2P 应用层组播

组播技术（Multicast）是一种针对多点传输和多方协作应用的组通信模型，有高效的数据传输效率，是下一代 Internet 应用的重要支撑技术。

早期的组播技术研究试图在 IP 层提供组播通信功能，但 IP 组播的实施涉及对现有网络基础设施的调整，因此，大规模应用受到限制。

随着 P2P 研究的兴起，基于应用层的组播技术逐渐受到广泛关注。应用层组播协议将组成员结点自组织成重叠网络（Overlay network），在主机结点实现组播功能，为数据多点并发传输提供服务。

应用层组播是在应用层实现组播功能而不需要网络层的支持，这样就可以避免出现由于网络层迟迟不能部署对组播的支持而使组播应用难以进行的情况。当然，应用层组播也有许多局限：

- ❑ 一是端系统对 IP 网络的了解有限，结点参与组网时，只能通过探测获得一些网络性能参数，选取的逻辑链路难以优化。
- ❑ 二是主机不了解 IP 网络的拓扑结构，只能通过带宽和时延等外在的特性参数，以启发式的方式建立重叠网络，逻辑链路不能较好地利用质量较好的底层网络资源，重叠网络的多条链路可能经过同一条物理链路。

4.11.3　网络游戏平台

大型网络在线游戏和网络对战游戏是不少“网虫”的至爱。但由于服务器能力有限，大型网络在线游戏往往需要限制场景人数或者不断增加服务器，而网络对战游戏也必须局限在局域网内进行或者依赖独立的服务器端程序及机器实现 Internet 上的电子竞技。目前，已有研究人员将 P2P 技术引入网络游戏和网络游戏支撑平台中。

目前较为成功的 P2P 游戏平台是华中科技大学集群与网格计算湖北省重点实验室推出的 PKTown 系统。PKTown 系统是一个支持多种网络对战游戏的 P2P 平台。P2P 网络对战游戏平台的难点在于将严格延时约束的结点聚集在一起，这由对战游戏本身要求所决定：延时是影响对战游戏用户体验的关键因素。在众多在线用户中，如何将新加入用户调度到周围都是延时邻近的环境中去呢？PKTown 也是采取 GridCast 中出现过的指数增长的同心圆环方式，很好地解决了这个问题。

PKTown 不需要改变游戏本身的代码，而是将用户和 Internet 邻居组建成一个虚拟局域网，将游戏发出的通信包截获后负载上虚拟局域网的地址，转发出去，游戏进程接收到之后认为是来自同一局域网的游戏包，则可以正常进行游戏。目前 PKTown 支持魔兽争霸、

星际争霸和反恐精英几款游戏，已经在高校范围内进行公测，并成功举办华中科技大学第三届 Race War 游戏大赛，用户反应良好。

自 P2P 技术从 1999 年出现之后，现在已经发展繁荣起来。前文中提到的很多技术都已经趋近成熟，如拓扑构建和内容分发等相关技术。由于 P2P 架构灵活，适用面广阔，所以将 P2P 应用到新领域的现象层出不穷，P2P 的软件产品也会更多地出现在人们的视野中。

4.12 P2P 系统应用及研发的平台

随着一些中间件，如 Java 虚拟机、网络浏览器和服务器的出现，应用对于操作系统环境的依赖越来越少。将来的系统有可能更多地依赖于平台来对用户和服务进行管理。P2P 平台可以支持基本的 P2P 模块功能，包括命名、查询、通信、安全以及资源整合等，它们对于操作系统的依赖较小。

目前一些主流的技术平台都支持端对端技术，包括 Sun 的 JXTA（Juxtaposition 的缩写）和微软的.NET 平台。事实上这样的平台越来越多，像 Next Page 正在使用点对点的技术以支持分布式的内容管理，Affiniti 则致力于研究面向对象的端对端的消息发送。

4.12.1 基于 JXTA 的 P2P 平台

JXTA 项目是由 Sun 公司研发的，旨在提供一个开放式、能够支持多种类型的分布式应用的平台。由于通信、安全、性能等方面的原因，JXTA 系统将所有的对等体分成若干个小组，这些对等体组（PeerGroup）是 JXTA 体系结构的核心。每一个对等体可以同时加入多个对等体组。JXTA 为对等体的查询、组成员管理提供核心协议，同时利用称之为 pipes 的异步单向通信信道来发送和接受消息。JXTA 中的数据都是以 XML 格式进行交换的。

JXTA 平台中，其 super-peer 结点中的路由结点是提供防火墙和 NAT 穿越功能的特殊结点。若通信一方在防火墙之后，那么单向穿越的过程由试图与防火墙后的 PeerA 连接的 peer 结点 B 发起，结点 B 首先与路由 Peer 连接，同时 PeerA 周期性地连接路由 Peer。由于路由 Peer 对 Peer A 是可见的，且不在防火墙后，连接请求消息可作为 http 的应答内容返回至 PeerA。若通信的双方都在防火墙之后，那么需要两个路由 peer。Peer1 通过路由结点 1 转发消息，路由结点 1 将消息转发至路由结点 2，Peer2 周期性地向路由结点 2 发送 http 请求，获得消息后断开连接。JXTA 平台中路由结点的使用不仅可以穿越防火墙，还可以解决瓶颈问题，解决网络传输的不兼容问题。

4.12.2 基于 DOTNet 的 P2P 平台

.NETMy Service 和.NET 平台的设计目的是能够使用户利用现存的一些标准，如 XML、UDDI、SOAP、WSDL 等访问 Internet 上的众多的资源和服务。.NET 采用一种新的程序语言 C#编写，其设计是集中围绕分布式服务的非集中性和非模块性进行的。

4.13　P2P 技术在企业级的应用

现在许多公司机构日益分散，给员工和客户提供轻松、方便的消息和协作的工具，变得日益重要。网络的出现，使协同工作成为可能。但传统的 Web 方式实现，给服务器带来了极大的负担，造成了昂贵的成本支出。P2P 技术的出现，使得互联网上任意两台计算机间都可建立实时的联系，建立一个安全、共享的虚拟空间，人们可以进行各种各样的活动，这些活动可以是同时进行，也可以交互进行。

4.13.1　企业管理

利用 P2P 技术可以进行有效的企业管理，它在企业管理方面的应用主要表现在下面几个方面。

- 协作：像 Gorove 软件这样的应用软件，可使企业雇员建立虚拟工作区，以供个人共享时间表和文档，进行声音、视频以及文本对话，并完成其他生产任务。
- 内容分配：P2P 软件通过允许系统在本地网上从本地的其他系统中搜索文件而令企业降低了场人 N 通信量，并使下载速度更快。
- 知识管理：P2P 知识管理软件使用智能代理对信息、Web 网站以及其他数据源进行审查，从而简化了信息处理。

4.13.2　电子商务

P2P 可为电子商务增加新功能，包括连接和实现各种链路。在原始结点上建立中心目录和搜索功能，可更有效地发送和反馈信息。它在电子商务上的应用，重点表现在以下两个方面。

1．金融服务

由于 P2P 的沟通只单纯涉及沟通的双方，不会有第三者知道双方沟通的信息，所以 P2P 非常适合发展在线金融服务。美国的 Billpoini 公司已将 P2P 技术应用于电子商务的付费机制，在 eBay 上，就向全球 35 个国家的使用者提供了这种技术，他们可直接用彼此的信用卡进行交易。

2．电子商务集市

利用 P2P 把庞大的文件交换社群转化为另类的电子商务集市，一家名为 Lightshare 的公司将推出一种服务，让电脑使用者直接通过其电脑销售数字产品，而不用经由 eBay 或亚马逊（Amazon.ocm）的中央服务器，这个服务从 eBay 脱胎，转化成点对点模式。

通过以上的描述可以看出，P2P 的基本原理是容易实现的，基于这一原理的应用各式各样，应有尽有。从 P2P 的发展来看，人们对 P2P 的研究方正在由基础架构的构建和维护及优化算法等桎梏中摆脱出来，开始深入到 P2P 技术的根本性问题中去。最新的 P2P 技术，

包括覆盖层网络的结点延时聚集、覆盖网之间（Inter-Overlay）优化、P2P 支撑平台以及 P2P 安全等方面。相信随着对 P2P 技术研究的不断深入，人们能够对 P2P 计算有一个更深入的认识并解决目前 P2P 领域中大部分科学问题。可以预见，P2P 所带来的技术创新和应用创新还将继续。

4.14　P2P 的影响及价值

P2P 技术的应用，不仅仅是目前所带来的新的信息共享方式，更重要的是 P2P 技术背后所代表的思想，即“非中心化”的理念的推广。这种非中心化的理念，将前所未有的刺激用户参与互联网的热情，过去位于网络中心的服务器作用将不断弱化，而位于网络边缘的资源重要性不断加强。网络结点之间的直接交流将成为交流方式的主体，不同网络终端之间的连接也不再困难，真正的网络融合成为可能，从而极大地丰富了网络的可用性、易用性，可以说 P2P 技术对整个互联网有着重要的影响及价值。

4.14.1　P2P 对资源信息生产的影响

在目前的互联网中，网络中存在着大量的“网络孤岛”。公司内部的局域网—Intranet 就是一个很好的例子。这些 Intranet 和 Internet 之间保持着一种若即若离的关系，一方面，它们需要连接到 Internet 上以获取丰富的资源；另一方面，它们又利用防火墙和代理服务器等手段将自己从 Internet 上隔离出去，同时还利用 DHCP（Dynamic Host Configuration Protocol，动态主机配置协议）和 NAT（Network Address Translate，网络地址转换）等技术将自己隐藏起来，从而形成“网络孤岛”。

因此，我们可以看出 Internet 中的设备总是从公用网络上获益，而它们中的大多数并没有给网络提供什么有价值的东西。也就是说，这种通信是单向的，而 P2P 技术则可以突破这种限制。

P2P 作为网络计算的一种新技术，它的目的是将网络中不同的计算机连接在一起，并充分利用互联网和 Web 站点中任何地方的闲置资源。P2P 技术大大方方地绕过了网址和各类解析服务器，让信息源和接收源可以形成独立的连接。它以用户为中心，所有的用户都是平等的伙伴。相隔万里的用户可以通过 P2P 共享硬盘上的文件、目录乃至整个硬盘和其他相关的网络资源。所有人都共享了他们认为最有价值的东西，这将使互联网上信息的价值得到极大提升。这种用户间直接交流的方式，使用户真正地参与到网络中，它改变了互联网现有的游戏规则，提升了整个网络的价值。边缘结点的参与，也使得信息生产点越来越多。

4.14.2　P2P 对信息传播的影响

P2P 技术可以被广泛应用于网络互联技术领域，并极大地提高对因特网中信息、带宽和计算资源的利用率。在传统的客户端/服务器模式下，虽然网络带宽成倍的增长，但是一方面热门的站点不堪重负，另一方面其他空闲的链路却白白地浪费掉了。利用 P2P 提供的

分布式结构可以有效的均衡负载，充分利用带宽。

利用 P2P 技术的信息产品流通方式，一方面利用了过去多余的带宽；另一方面信息生产者和最终用户之间进行的交易，节省了传统方式中信息产品的流通成本，比如相关载体的制作、宣传、物流和中介费用，使信息复制的边际成本趋向于 0。

因此，利用 P2P 网络传输这种传播方式迟早有一天会取代传统的以磁带、光盘为载体的影视音乐发行渠道，从而成为人们获取影音资源的主要渠道。当然，这个进程的长短要看国内外环境的不同而定。

4.14.3　P2P 对资源交互的影响

在传统的 Web 方式中，要实现文件交换，就需要网络服务器的参与，即首先要将文件上传到某个特定的专业下载网站 A，其他用户则通过另外一个网站 B 的搜索引擎搜索所需要的文件的信息，然后根据搜索引擎提供的 URL 连接到专业的下载网站 A 下载该文件。

这种方式在用户交换文件时非常不方便。而 P2P 技术可以让用户之间直接进行文件交换，没有任何的中间环节。P2P 软件用户可以将自己机器上的任何文件设置为共享；同时他也可以从其他的 P2P 软件用户的机器上下载自己所需要的文件。这样的信息共享方式使得互联网上的用户可以像局域网用户一样协同作业和自由交流，使每一台计算机的硬盘空间和数据信息可以被充分利用，使互联网中的信息交流、文件交换、协同工作和局域网中一样简单和方便。

P2P 搜索可以使用户更加快速、准确地找到他们想要的信息资源。而且，相对于传统的搜索方式，P2P 搜索的范围更大，得到的结果也就更多。这是因为 P2P 的搜索目标不仅有传统的 WWW 服务器，还包括网上难以数计的个人电脑。传统的搜索引擎只能搜索到 20%～30%的网络资源，而运用 P2P 技术的搜索引擎则可以在理论上搜索到网络上所有开放的信息资源。

P2P 技术带来的另一个变化就是改变了内容所在的位置，内容正在从“中心”走向“边缘”。就是说内容不是存在几个主要的服务器上，而是存在所有用户的电脑上。

总体来说，P2P 技术对整个互联网影响及其价值是不可估量的，也是一言难尽的。然而，P2P 对于用户最大的意义，不是它的技术和功能，而是它的理念。这种理念源于人们互联网的憧憬和梦想，它使网络回归到 Internet 的本质，让共享与自由的精神充满网络世界。中国 P2P 的发展，必将经历一个从技术到理念的过渡，技术的不断进步为中国 P2P 的发展铺平道路，而理念的不断更新，则为中国 P2P 的发展指明了方向。

4.15　P2P 应用所存在的问题

P2P 在得到广泛应用的同时，也引起了不少的争议。这些争议一方面妨碍了 P2P 业务发展，另一方面也在促使 PPZ 不断的进行自我的改进，以得到更好的发展。

4.15.1　版权问题

文件交换是目前 P2P 最为流行和普遍的应用。而文件交换的主要对象就是音频、视频

内容。因此，这些 P2P 应用就不可避免的会引起知识产权冲突。Napster 让人们开始关注 P2P，它庞大的客户量也同样引起了唱片公司的注意，并在与唱片公司的知识产权较量中败下阵来。不久前，香港一男子也因为通过 BT 网络发布电影种子而被判入狱。因此，版权争端是 P2P 发展过程中一个无法绕过的门槛。

4.15.2　带宽问题

P2P 技术为用户提供了丰富的资源，给用户带来了极大的便利，使得用户可以随时在网上找到自己所需要的资源。而且 P2P 应用的特性之一就是会使用的人越多，性能越高。但同时由于 P2P 网络规模的不断扩大和用户下载文件数量的增多，吞噬网络带宽的问题也日益突出。宽带提供商在重负之下收入却没有什么增长，自然也对此颇有微词，有的甚至不惜封杀 P2P 端口。

4.15.3　垃圾信息问题

由于 P2P 网络的用户众多，当某个用户进行搜索时，自然会得到大量的搜索结果。而除了少数有用的信息以外，其他大多数的信息可能都属于垃圾信息。在缺乏统一管理的情况下，P2P 网络很难对搜索结果进行排序，用户将不可避免地陷入垃圾信息的汪洋大海。现在已经有公司尝试着将人工智能技术、专家数据库技术引入 P2P 网络中，希望能够克服垃圾信息的困扰。

4.15.4　管理困难的问题

P2P 网络的特点就在于其无中心的等对方式，这种方式用户以极大的自由，但是也存在着“无政府主义”的困境。缺乏管理的 P2P 网络对于一些不良的信息和内容也失去了监控的机制。而且，由于在 P2P 网络中目前存在着的付费、流量计算、商品价值的验证等问题，通过 P2P 网络开展电子商务也很困难。

4.15.5　网络安全问题

在传统的 C/S 网络中，服务器起着中心结点的作用，为保证服务器的正常运转，各种防火墙、防病毒的技术无不用其极致，但仍然难以避免受到攻击而瘫痪。在 P2P 网络中，各网络结点都是普通的 PC 机，其防范病毒和黑客攻击的能力自然不能和服务器相比，结点受到攻击后会造成两种后果：

- 一是用户个人的信息被剽窃，文件被病毒感染，其他从该结点下载文件的用户也被感染。
- 二是用户站点被黑客控制，成为分布式拒绝服务的发起者。而且，在商业中使用这种共享的关键数据很可能导致严重的安全问题。

4.15.6　标准之争

P2P 技术目前并没有统一的技术标准和开发平台，但是由于其所蕴含的无限商机，P2P 技术自然就成为各大厂商争夺的焦点。Intel 是 P2P 的热心鼓吹者，并且试图以 P2P 开发组织盟主的身份去领导 P2P 的未来。可惜事情的进展并未遂 Intel 的愿，在 Intel 主持的 P2P 工作组首次会议上，上至 BIM、Sun、HP 这样的 TI 巨头，下至一些头天才冒出来的一些初创公司，没有人将 Intel 视作权威。与会者纷纷指责 Intel 的组织无方，呼吁 P2P 工作组应当参照 EITF（Internet 工程任务组）的管理模式，SUN 公司甚至“另开炉灶”，主张基于 Java 和 Jini 语言的 P2P 应用开发。利益的分歧导致开发标准的难以统一，这成为 P2P 发展道路上的另一个难迈的坎。

4.15.7　互操作性问题

P2P 系统必须面对不同的操作系统、网络和技术平台。现在的 P2P 应用执行的是相对简单的任务，例如传输 MP3 音乐文件，它们能够和脚本翻译、软件打包以及互联网上其他互操作性技术兼容。将来 P2P 系统需要更高级的互操作性技术来执行更为复杂的任务。

4.15.8　拓扑的一致性和资源定位

在缺少一个集中化服务器的动态环境下，各个结点能够维持一致的网络拓扑信息。由于 P2P 网络中结点的加入和离开非常的频繁，传统路由扩散的方式无法解决这一问题，所以需要一个高效的一致性信息维护机制来实现这些功能。另外，用户从大量的分散结点中找到需要的资源和服务也是一个挑战。

尽管 P2P 技术还存在着以上很多不足，也需要不断的改进和完美，但是 P2P 作为一种崭新的传输模式，在加强网络上人的交流、文件交换、分布式计算、服务共享等方面已经充分显示出了其强大的技术优势，在改进不足的基础上，必定会有更大的发展。

4.16　本 章 小 结

P2P 是刚崛起的一种新技术，它的出现正在深刻快速地影响互联网今后的架构：服务的提供者开始把网络的控制权完全交给用户，而用户与用户之间开始真正直接、简单、自由的沟通。

P2P 不仅蕴含了丰富的思想，也体现了一种崭新的网络技术。它成功地将网络资源进行整合与累计，通过网络架构技术、内容存储技术、内容传输及共享技术等，将数以亿计的 CPU 及数以百亿计的硬盘资源整合到互联网上，最大限度地开创一个平等的共享世界。同时，P2P 穿越了防火墙，绕过了 DNS 机制，创造了一个远离 DNS 的网络传输平台，可

以说 P2P 技术正在改变 Web 的存在，传统的互联网思想和技术正在变革。

P2P 在应用方面也有极其广泛的可应用空间，在文件下载、传输、分布式计算、即时通信及协同工作方面有着不可替代的作用。随着 P2P 研究的理一步深入，一定会涌现出更多更新更有创意的应用模式，必将为信息社会带来更好的机遇与挑战。

P2P 技术将改变当前信息产品的生产、流通、交易、消费等所有环节，将形成一次数字化产业的革命。总而言之，P2P 技术将颠覆下一代互联网游戏规则；P2P 技术在未来几年内一定会改写互联网、通信和广电领域的历史。

第 2 篇　技术应用篇

第 5 章　P2P 网络中的 NAT 穿透技术

目前的互联网是基于 IPv4 架构的，随着计算机接入数量的不断增加，IP 地址资源愈加匮乏。为了解决该问题，各企业、学校及 Internet 服务提供商都部署了大量 NAT（Network Address Translation，网络地址转换），通过 NAT 也就是网络地址转换技术来解决共享上网问题。

就 P2P 网络而言，它是构建在 Internet 网络之上的网络，Internet 网络的结构和特点与 P2P 网络的性能有着必然的联系，在 Internet 网络中，利用 NAT 技术，可以使一个局域网内的所有主机通过一个或几个公网 IP 地址来访问 Internet。但由于局域网与 Internet 编址方式的不同，NAT 设备掩藏了参与构建 P2P 网络的大量用户结点，这就阻碍了 P2P 网络结点相互之间正常的数据转发，那么大量 NAT 设备的存在就会阻碍 P2P 的应用普及和发展。

目前运用 NAT 穿透技术可以有效地解决以上问题，通过 NAT 穿透，可以穿越 NAT 设备和防火墙，实现公网与私有网络之间的连通，最大限度地发现 P2P 网络中所有参与的网络结点。本章就讲解一下 P2P 网络中的 NAT 穿透技术。本章主要讲述的基础知识如下。

- NAT 的基本概念：了解什么是 NAT，NAT 的背景知识、自身的优点及应用的环境等。
- NAT 的工作原理：重点掌握 NAT 的工作原理，掌握 NAT 工作的几种方式。
- NAT 的分类：了解 NAT 在应用中的分类，掌握不同 NAT 的工作方式和原理以及它们之间的区别。
- NAT 的穿透技术：NAT 的穿透是本章的重点，要重点掌握在 UDP 方式连接下的 NAT 穿透，了解相关 NAT 的穿透方法、穿透机制和实现模型，也要了解在 TCP 连接下的 NAT 穿透。
- NAT 穿透在 P2P 系统中的应用：要了解在实现的 P2P 应用系统中是如何应用 NAT 的穿透技术的。
- 最后要了解一种 UPnP 的 P2P 端口映射技术，了解它的工作原理、功能及如何设置并在 P2P 系统中应用 UPnP 的。

5.1　什么是 NAT

本节主要讲解 NAT 的基本概念，简要说明 NAT 的产生背景，对 NAT 的技术描述、应用优点、应用环境及工作配置都一一做了说明。

5.1.1　NAT 简介

NAT 技术指的是将私有的专用网络地址转换为公用网络地址的一种技术。

在一个内部局域网络中，主机的 IP 地址是可以在满足 IP 地址规则的前提下随意自定义的，各主机间通过内部的 IP 地址进行通信。而当内部的计算机要与外部 Internet 网络进行通信时，就需要借助于有 NAT 功能的设备，如路由器等，它负责将其内部的 IP 地址转换为合法的公网 IP 地址以实现正常的通信功能。在这一过程中所用到的与网络地址转换相关的技术就是 NAT 技术。

5.1.2　NAT 的产生背景

随着 Internet 的发展和网络应用的增多，IPv4 地址枯竭已成为制约网络发展的瓶颈。尽管 IPv6 可以从根本上解决 IPv4 地址空间不足问题，但目前众多网络设备和网络应用大多是基于 IPv4 的，因此在 IPv6 广泛应用之前，一些过渡技术（如 CIDR、私有网络地址等）的使用是解决这个问题最主要的技术手段。

其中，使用私有网络地址之所以能够节省 IPv4 地址，主要是利用了这样一个事实：一个局域网中在一定时间内只有很少的主机需要访问外部网络，而 80%左右的流量只局限于局域网内部。由于局域网内部的互访可通过本地私有网地址实现，且私有网络地址在不同局域网内可被重复利用，因此可有效缓解 IPv4 地址不足的问题。当局域网内的主机要访问外部网络时，只需通过 NAT 技术将其私有网络地址转换为公网地址即可，这样既可保证网络互通，又节省了公网地址。

5.1.3　NAT 技术的优点

作为一种过渡方案，NAT 通过地址重用的方法来满足 IP 地址的需要，可以在一定程度上缓解 IP 地址空间枯竭的压力。它具备以下优点。

- 共享上网：NAT（网络地址转换）提供了局域网共享上网的简单方案，内部网络用户连接互联网时，NAT 将用户的内部 IP 地址转换成一个外部公共的 IP 地址。当数据从外部返回时，NAT 反向将目标地址替换成初始的内部用户地址，这样可以用少量公网 IP 通过 NAT 技术实现大量的共享上网 IP。
- 天然防火墙：通过静态映射，不同的内部服务器可以映射到同一个公网地址。外部用户可通过公网地址和端口访问不同的内部服务器，在与外部 Internet 进行通信的过程中，可以隐藏内部 IP 地址，同时也隐藏内部的网络结构，从而防止外部对内部服务器乃至内部网络的攻击行为。降低了内部网络受到攻击的风险，构成了一个天然的防火墙。
- 方便网络管理：如通过改变映射表就可实现私有网络服务器的迁移，内部网络的改变也很容易，这样便于对网络的管理。
- 节约公网 IP：NAT 技术对外隐藏了内部管理的 IP 地址。这样，通过在内部使用非注册的 IP 地址，并将它们转换为一小部分外部注册的 IP 地址，从而减少了 IP 地址注册的费用，节省了目前越来越缺乏的地址空间（即 IPv4）。

5.1.4　NAT 的应用环境

NAT 的应用也是有环境要求的，从 NAT 的实际应用情况来看，常在以下的网络环境

中使用 NAT 技术。

- 一个内部的网络环境不想让外部网络用户知道自己的网络内部结构，可以通过 NAT 将内部网络与外部 Internet 隔离开，则外部用户根本不知道通过 NAT 设置的内部 IP 地址。
- 在一个网络环境中，合法 Internet IP 地址很少，而内部网络用户很多。可以通过 NAT 功能实现多个用户同时公用一个合法 IP 与外部 Internet 进行通信。

5.1.5　NAT 工作环境的软硬件配置

NAT 的工作可以在很多设备上实现，像防火墙，路由器或者计算机等。这些设备都是本地网络和 Internet 网络的边界，要构建一个 NAT 工作环境，基本需要以下几方面的配置。

- 在硬件上，至少需要一个具备 NAT 功能的路由器；
- 软件上，需要配置在路由上的能实现 NAT 功能的 IOS；
- 在配置过程上，路由器至少要有一个内部端口（Inside），一个外部端口（Outside）。

内部端口连接的网络用户使用的是内部 IP 地址，内部端口可以为任意一个路由器端口。外部端口连接的是外部的网络，如 Internet，外部端口可以为路由器上的任意端口。

5.2　NAT 的工作原理

在 5.1 节中已经知道了 NAT 的基本概念，要想在实际的网络环境中对 NAT 进行穿透，就必需了解它的工作原理，只有知道了 NAT 的工作方法、实现原理，才能对症下药，找到正确的穿透 NAT 的方法。本节就讲一下 NAT 的工作原理。

5.2.1　NAT 技术相关的几个概念

在 NAT 技术中，首先要明白几个概念，1 个内部网的网络环境和 4 个地址术语是必须正确理解的，网络环境指的是内部网络，4 个地址术语分别指的是 Inside Local、Inside Global、OutsideLocal 和 Outside Global。

1．关于内部网

在 NAT 技术中所说的内部网，通常是一个局域网或 LAN，一般是一个孤立的域。在这个网络域中，所有的 IP 地址都必须保持唯一性，当这个孤立域的设备需要访问外网或 Internet 时，需要使用 NAT 技术将孤立域的 IP 地址翻译成能在公网上使用的，具有唯一性的地址，所有具备这些特点的局域网络都叫做内部网。

2．4个地址术语

在 4 个地址术语中，Inside（内部）是指那些由机构或企业所拥有的内部网络，这些网络上的主机通常分配了私有地址。这些地址不能直接在 Internet 上进行路由，从而也就不

能直接用于对 Internet 的访问，必须通过网络地址的转换，以合法 IP 的身份来访问 Internet。前者即 Inside Local 地址。后者则为 Inside Global 地址。Local（本地）的地址是不能在 Internet 上通信的 IP 地址；Global（全局）的地址是能在 Internet 上通信的地址；Outside（外部）是指除了所考察的内部网络之外的所有网络，主要指 Internet 万维网。

有了对 Inside、Outsider、Local 和 Global4 个词的解释，再看一下 4 个地址的定义。

- Inside Local Address（内部本地地址）：指一个网络内部分配给网上主机的 IP 地址，此地址通常不是 Internet 上的合法地址，即不是网络信息中心（NIC）或 Internet 服务提供商（ISP）所分配的 IP 地址。
- Inside Global Address（内部全局地址）：用来代替一个或者多个内部本地 IP 地址的、对外的、Internet 上合法的 IP 地址。
- Outside Local Address（外部本地地址）：一个外部主机相对于内部网所用的 IP 地址。此地址需要是 Internet 上合法的地址，但是从内部网可以进行路由的地址空间中进行分配的。
- Outside Global Address（外部全局地址）：由主机拥有者分配给在外部网上主机的 IP 地址。此地址是从一个从全局可路由的地址或网络空间中分配的。

5.2.2　NAT 的工作原理

NAT 在工作过程中，根据数据包信息发送和请求的方式不同，可分为 3 种不同的情况进行相对应的处理工作。

（1）当孤立的 LAN 中的机器需要相互访问时，通常它们只需要使用 inside local address 直接进行通信。不需要地址翻译。

（2）当 LAN 中的机器需要访问外部的网络时，数据包首先会被送到该机器所在的网关，这个网关除了路由数据包，还有地址翻译的任务。当网关收到数据包并确定可以路由后，会检查该包是否符合 NAT 的标准，然后会给 inside local address 找一个 inside global address。于是，数据包的源地址被翻译成 inside global address 并被送到目的地。

（3）当公网上的机器要送一个包给内网机器时，数据包的目的地址是 inside global address。当数据包到达内网的网关时，网关同样要做地址翻译，将 inside global address 翻译成 inside local address，然后将数据包送给在内部网的目的机器。

NAT 的工作原理，可用图 5.1 表示。

根据图 5.1 所示的原理，可以用下面的通信实例来进一步阐述 NAT 的工作过程。以笔者所在的局域网为例，配置上有一个具备 NAT 功能的路由器用来作为与外网通信的接口，用一个单一的公网 IP 进行共享上网，在内部网络中使用 192.168.1.1～192.168.1.255 作为其网络内部的私有 IP，Internet 网络服务提供商为此内部局域网分配的单一公网 IP 为 123.118.158.131。

此时，当局域网内部一台内部私有地址为 192.168.1.20 的主机访问 IP 地址为 220.181.6.18 的网站服务器时，其整个通信过程如下。

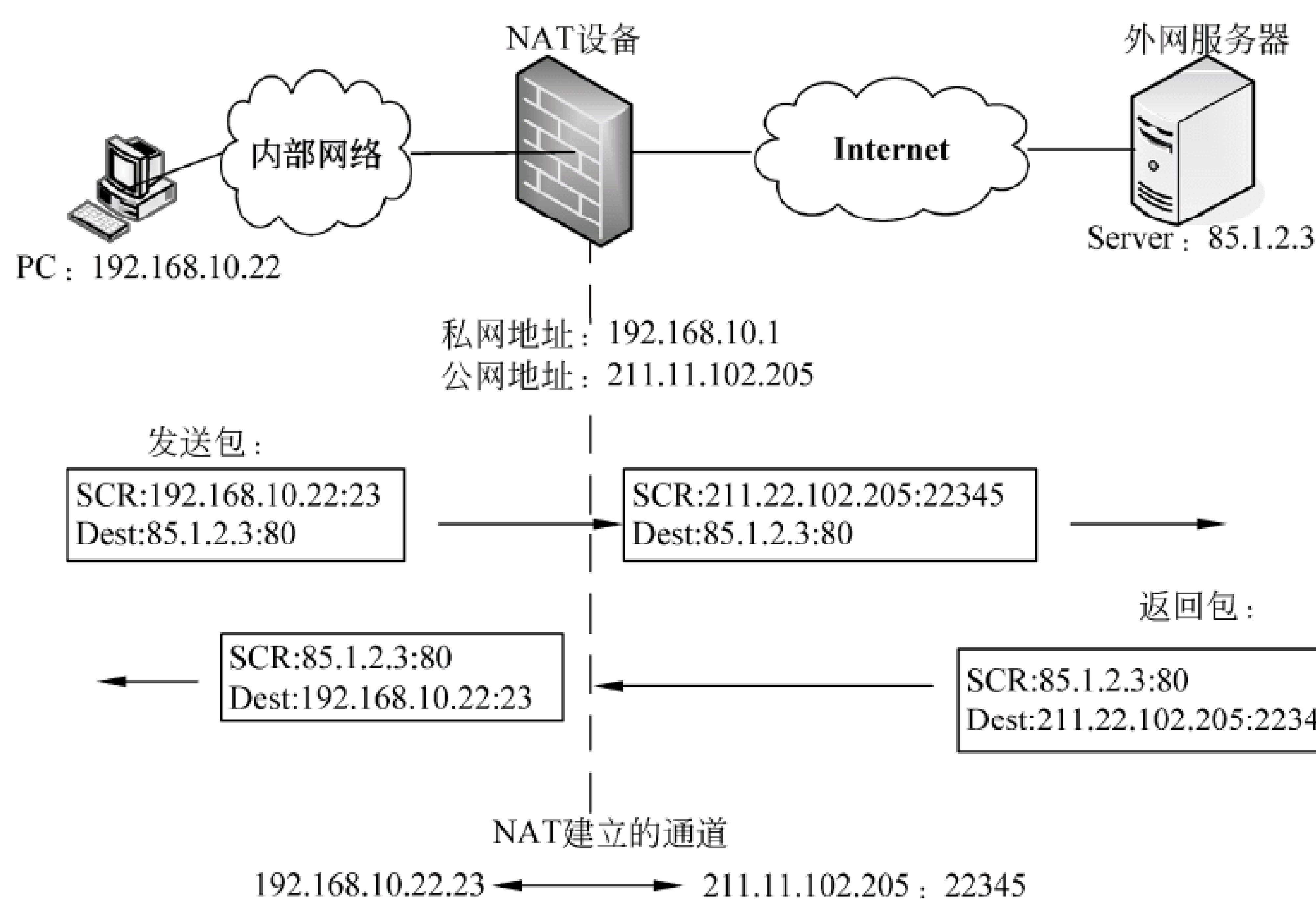


图 5.1　NAT 的工作原理图

NAT 的工作原理实例

（1）IP 地址为：192.168.1.20 的主机通过 TCP/IP 协议产生一个包含以下在 IP 和 TCP 或者 UDP 标头中的数值的 IP 数据包：

- 目标 IP 地址：220.181.6.18；
- 源 IP 地址：192.168.1.20；
- 目标端口：TCP 端口 80；
- 源端口：TCP 端口 2000。

（2）内部网中发出请求的源主机将此 IP 数据包发送给本网络中的具有 NAT 功能的路由器或其他 NAT 设备，NAT 设备接收到此数据包后，将此数据包解析后向公网上发送如下的数据包信息。

- 目标 IP 地址：220.181.6.18；
- 源 IP 地址：123.118.158.131；
- 目标端口：TCP 端口 80；
- 源端口：TCP 端口 5000。

（3）NAT 将重新映射后的 IP 数据包发送到 Internet。IP 地址为 220.181.6.18 的网站服务器在接收到请求信息后，向局域网中的 NAT 设备返回一个响应，响应信息的数据包信息如下。

- 目标 IP 地址：123.118.158.131；
- 源 IP 地址：220.181.6.18；
- 目标端口：TCP 端口 5000；
- 源端口：TCP 端口 80。

（4）当局域网中的 NAT 接收到外网服务发来的响应信息后，开始对此响应信息进行解析。在完成解析和相应地址的映射后，它将此数据包发送给局域网中的 IP 地址为 192.168.1.20 的 Internet 请求客户端，数据包的信息如下。

- 目标 IP 地址：192.168.0.99；
- 源 IP 地址：157.60.0.1；
- 目标端口：TCP 端口 2000；
- 源端口：TCP 端口 80。

（5）对于来自 NAT 协议的传出数据包，源 IP 地址（专用地址）被映射到公用地址，并且 TCP/UDP 端口号也会被映射到不同的 TCP/UDP 端口号。

5.2.3　NAT 的工作方式

NAT 在进行网络地址转换的时候，有以下几种工作方式。

- 静态 NAT：将一个私有网络地址和一个公共 IP 地址做一对一映射，如果内网的机器需要被外网访问，这种方式非常有用。
- 动态 NAT：将一个私有网络地址和一个公共地址池中的某个 IP 地址做映射。在映射关系建立后，也是一对一的地址映射，但所使用的公网 IP 地址不确定。
- overloading：是一种特殊的动态 NAT，将多个私有网络 IP 地址映射到一个公网 IP 地址的不同端口号下，通常也称之为 PAT（Port Address Translation）。
- overlapping：这种情况比较复杂，指内网所使用的 IP 地址与公网（internet）上的地址有重合。这时，路由器必须维护一个表：在数据包流向公网时，它能够将内网中的 IP 地址翻译成一个公网地址；在数据包流向内网时，把公网中的 IP 地址翻译成与内网不重复的地址。

5.3　NAT 的分类

网络地址转换，也就是 NAT 有多种类别，每种类别按其性质、特点、应用及功能又分为多个不同的子类别。但总体上看，NAT 就可以简单的分为两种主要的类型，分别为 Basic NAT 和 NAPT，它们的分类关系如图 5.2 所示。

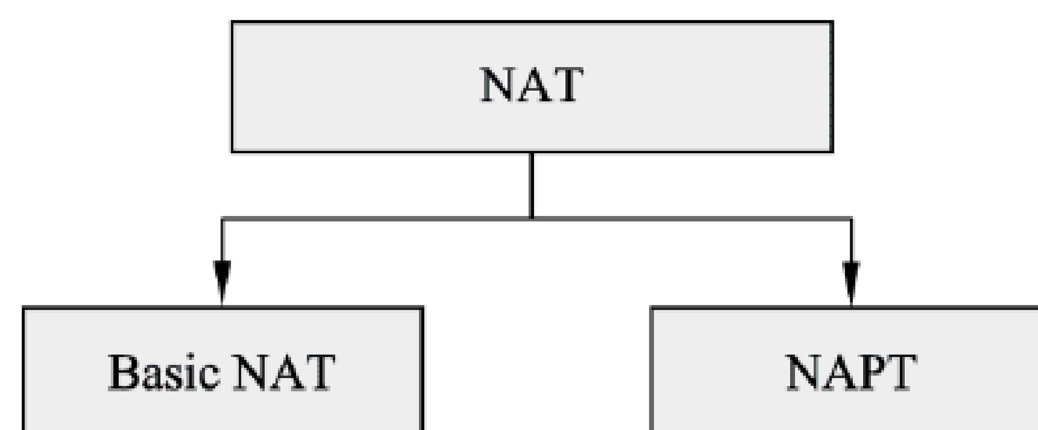


图 5.2　NAT 的基本分类

本节就重点讲解 NAT 的分类。图 5.2 所示的 NAT 所属的两个大的类别下，还细分有其他相应的子类别，如下就是对 NAT 分类的具体说明。

5.3.1　基本 NAT（Basic NAT）

Basic NAT（全称为 Basic Network Address Translation），也就是通常所说的基本 NAT。

Basic NAT 是最先被提出来的，它的工作方式属于一对一的地址转换，在这种方式下只转换 IP 地址，而对 TCP/UDP 协议的端口号不处理。当内网的主机需要访问外网时，它将内部主机的私有 IP 地址转化为全球唯一的 IP 地址。

只有 Basic NAT 有很多可用的公网 IP 地址时，它才有作用，才能把地址绑定在内部主机上。Basic NAT 在工作的过程中主要完成以下几个步骤。

（1）NAT 设备收到私有网络主机发送的访问公网服务器的报文。

（2）NAT 设备从地址池中选取一个空闲的公网 IP 地址，建立与私有网络主机报文源 IP 地址间的 NAT 转换表项（正反向），并依据查找正向 NAT 表项的结果将报文转换后向公网的一端发送。

（3）NAT 设备收到公网的回应报文后，根据其目的 IP 地址查找反向 NAT 表项，并依据查表结果将报文转换后向私有网络发送。

5.3.2　网络地址及端口转换（NAPT）

NAPT（全称是 Network Address/Port Translator），网络地址端口翻译，是把“基本 NAT”翻译的概念延伸了一步。在翻译地址的同时也翻译传输层标志，如 TCP/UDP 的端口号、ICMP 的查询 ID 等信息，从而把多个内部主机的传输层标志复用为一个唯一的外部地址。它允许内网的多个主机对应一个全球唯一的 IP，这样 NAPT 使得一组主机可以共享一个唯一的外部地址。在实际的组网使用过程中可以把 NAPT 和 Basic NAT 结合起来，这样就可以将一组外部地址与端口的翻译关联起来。

NAPT 对访问请求的处理是这样的，对于从内部网向外的访问请求，NAPT 将翻译源 IP 地址、源传输层标志以及相关的字段，与源地址传输层标志相关的字段包括 IP、TCP、UDP 和 ICMP 头校验和等信息，这些信息也都在 NAPT 的翻译之列。对于由 Internet 网络中进入内部网的数据包，NAPT 将翻译目的 IP 地址、目的传输层标志以及 IP 层和传输层头校验和信息。传输层标志可以是 TCP/UDP 端口号或 ICMP 查询 ID 中的任意一种。NAPT 的工作过程可由以下几步来完成。

（1）内部网中的某一主机向公网发送服务请求的时候，NAPT 设备将接收到内部网发送的访问请求的报文数据。

（2）NAT 设备从地址池中选取一对空闲的“公网 IP 地址：端口号”对，建立与内部网发送的报文“源 IP 地址：源端口号”之间的 NAPT 转换表，并依据查找正向 NAPT 表项的结果将报文转换后向公网发送。

（3）公网服务器在接收到请求报文后，将对此请求产生一个应答报文，NAT 设备收到公网的应答报文后，根据其“目的 IP 地址：目的端口号”查找反向 NAPT 表，并依据查表结果将报文转换后向内部网主机发送。

以上是 NAPT 的完整工作过程。需要注意的是，NAPT 转换表有正向和反向之分，会根据通信形式来决定是查找正向还是反向表。

在 NAT 的两大主要类别中，Basic NAT 是 20 世纪 90 年代中期提出来的，现在用的已经很少。就目前的应用来看，现在大多数 NAT 设备都是 NAPT 类型的，NAPT 在整个 NAT 的设备和使用中，占了大部分的比例，下面详细讲解一下 NAPT。

NAPT 是一类 NAT 设备的总称，它又可以分为两种类型，分别为对称 NAT（Symmetric

NAT）和克隆 NAT（Clone NAT），它们的关系如图 5.3 所示。

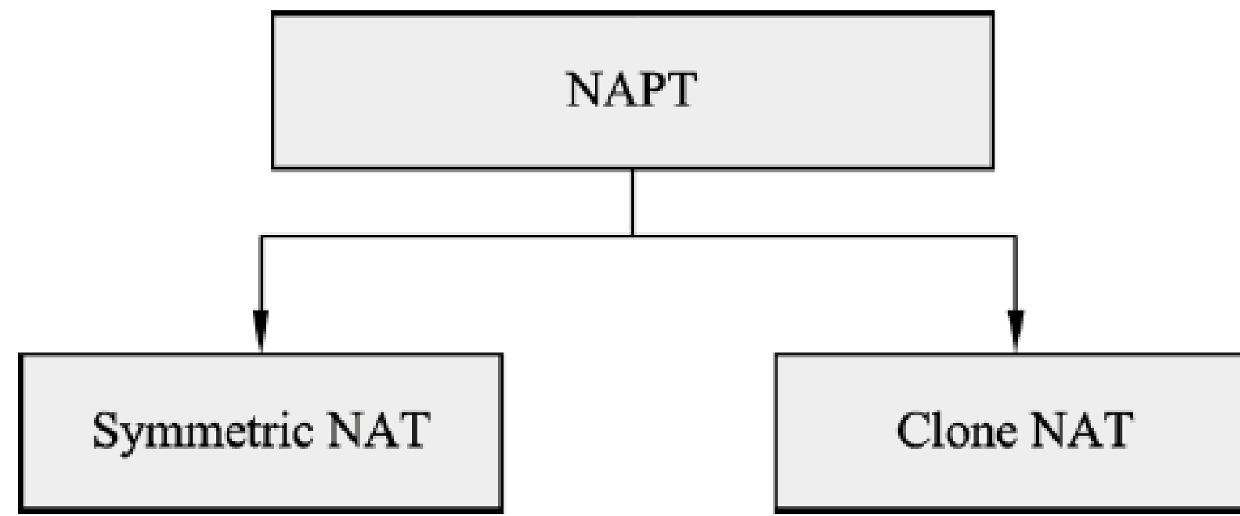


图 5.3　NAPT 的基本分类图

5.3.3　对称 NAT（Symmetric NAT）

对称式 NAT（Symmetric NAT），是这样工作的，指把所有来自相同内部网的 IP 地址和端口号，到特定目的 IP 地址和端口号的请求映射到相同的外部 IP 地址和端口。如果同一主机使用不同的源地址和端口，发送的目的地址不同，则使用不同的映射。只有收到了一个 IP 包的外部主机才能够向该内部主机发送回一个 UDP 包。

对称式的 NAT 不保证所有会话中的（私有地址，私有端口）和（公开 IP，公开端口）之间绑定的一致性。相反，它为每个新的会话分配一个新的端口号。

从图 5.4 中可以看出，假如客户 A 分别从相同的内部地址和端口号（192.168.1.20：1234）同时发起两个会话请求到 Server 1 和 Server 2，对称 NAT 可能会为这两个来自相同地点的会话请求分配不相同的公开端点号。

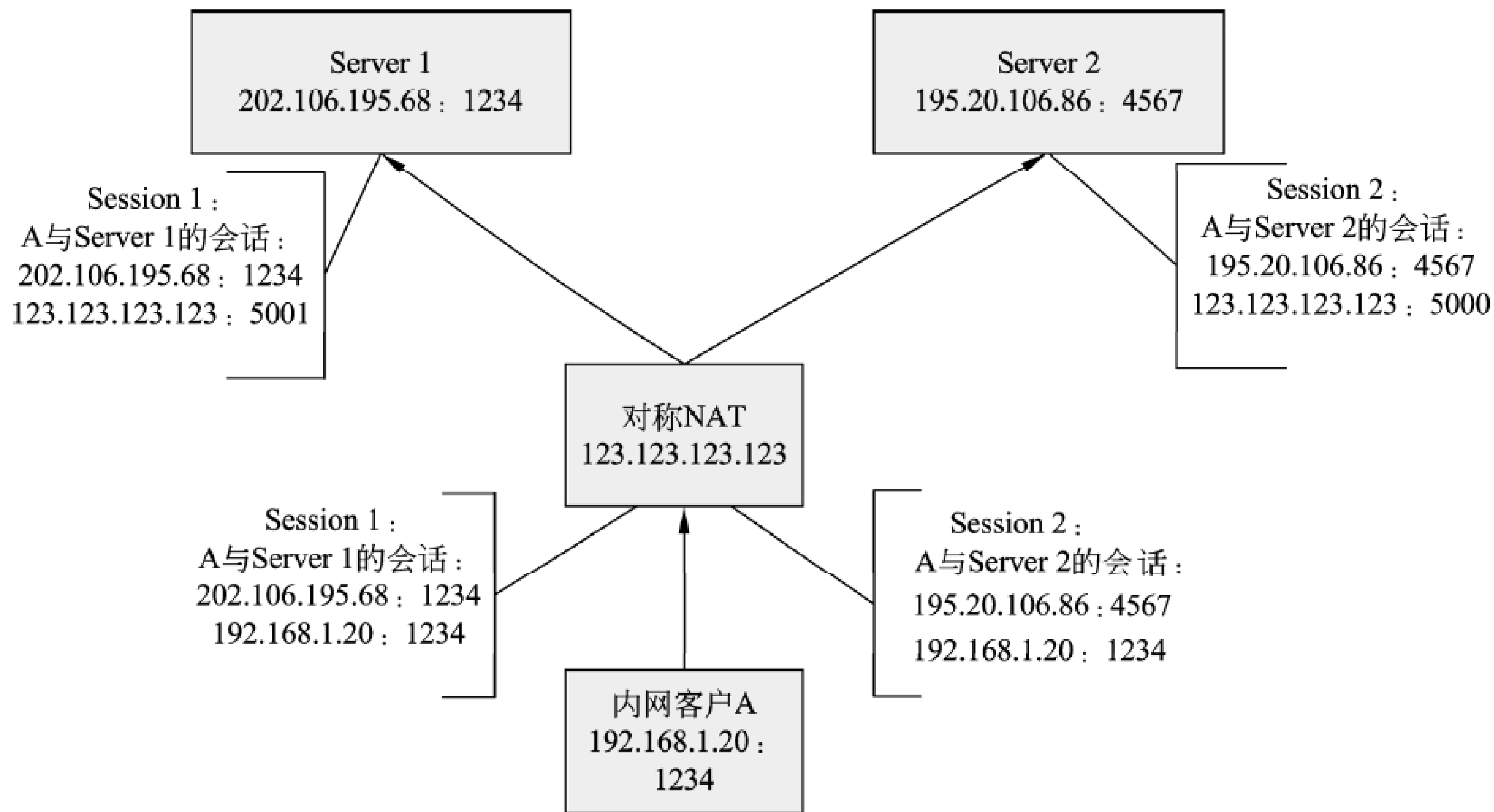


图 5.4　对称 NAT 的工作原理

图 5.4 中，对称 NAT 将 123.123.123.123：5001 分配给会话 1，把 123.123.123.123：5000 分配给会话 2。因为这两个会话有一个端点不同，所以虽然在翻译的过程中客户 A 的身份发生了变化，但 NAT 仍然能够正确工作。

5.3.4　Clone NAT　（克隆 NAT）

当在（私有 IP，私有端口）与（公网 IP，公开端口）中已经建立了一个端口映射表后，克隆 NAT 将为随后从相同的私有地址和端口号发起的呼叫重复使用该映射。条件是只要使用映射（或称这种映射为绑定）的会话，至少有一个继续保持激活状态，如图 5.5 所示为 Clone NAT 的工作原理。

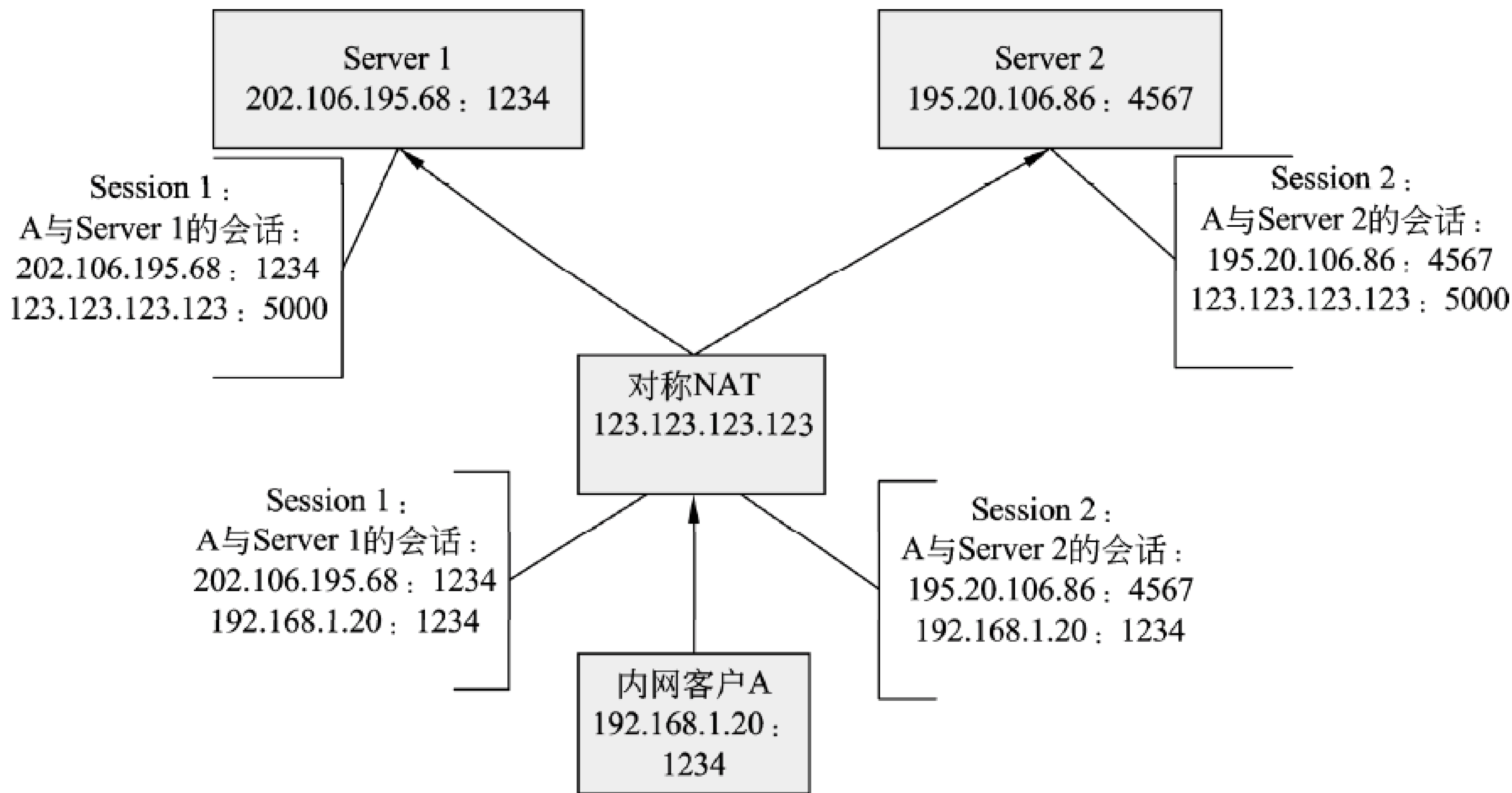


图 5.5　Clone NAT 的工作原理

从图 5.5 中可以看出，客户 A 分别从相同的内部地址和端口号（192.168.1.20：1234）同时发起两个会话请求到 Server 1 和 Server 2。因为这两个请求来自相同的内部地址和端口，所以克隆 NAT 将为这两个不同的会话请求分配相同的公开端点号（123.123.123.123：5000），以保证客户 A 的“身份”能够在经过翻译后仍然保持一致。NAT 和防火墙不翻译端口号，因此也称这种工作方式的 NAT 为克隆方式的 NAT。

根据克隆时受到的限制大小，又可以把克隆 NAT 分为 3 种类型，分别为 Full Clone、Restricted Clone、Port Restricted Clone，它们之间的关系如图 5.6 所示。

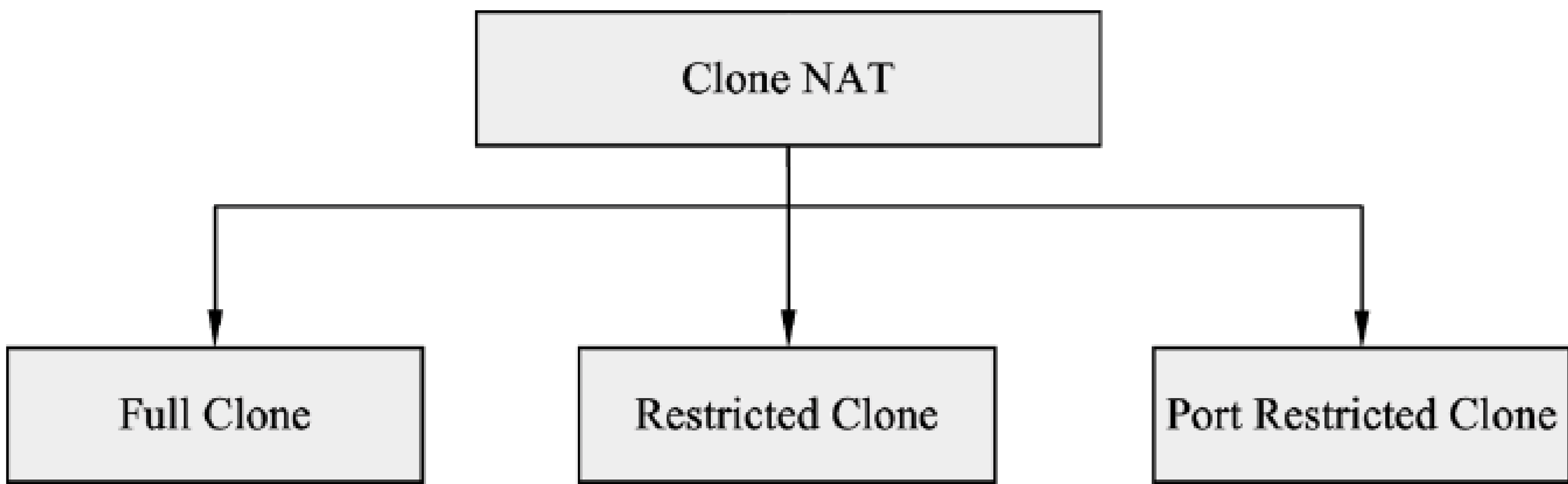


图 5.6　Clone NAT 的分类图

1．Full Clone（全克隆）

Full Clone NAT，首先把所有来自相同内部 IP 地址和端口的请求映射到相同的外部 1P

地址和端口。其次，任何一个外部主机通过把一个 TP 包发送给已得到映射的外部 IP 地址的方式，都能够把该包发送给内部主机。

2．Restricted Clone（限制性克隆）

Restricted Clone NAT，把所有来自相同内部 IP 地址和端口号的请求映射到相同的外部 IP 地址和端口。与全克隆 NAT 方式不同，只有当内部主机以前曾经给 IP 地址为 x 的外部主机发送过一个包时，IP 地址为 x 的该外部主机才能够把一个 IP 包发送给该内部主机。

3．Port Restricted Clone（端口限制性克隆）

Port Restricted Clone NAT，与限制性克隆类似，只是限制中多了端口号。特别是，一个外部主机可以发送一个源 IP 地址和源端口号分别为（x，P）的 IP 包给外部主机，只有当内部主机以前曾经给 IP 地址为 x，端口号为 P 的外部主机发送过一个包时，IP 地址为 x 的该外部主机才能够把一个源端口号为 P 的 IP 包发送给该内部主机。

综上所述，整个 NAT 体系的分类，可以用下图 5.7 完整地表示出来。

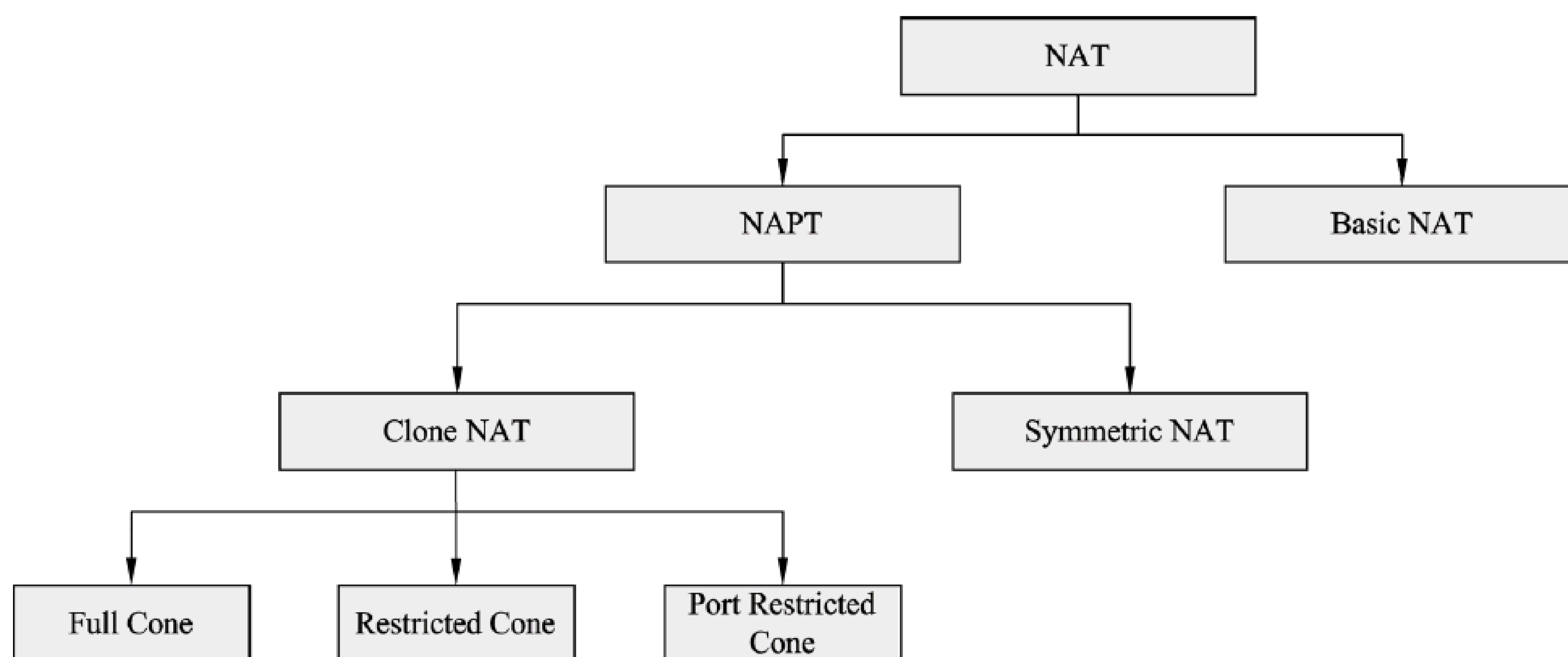


图 5.7　NAT 体系全分类图

5.4　UDP 连接下的 NAT 穿透技术

目前的互联网是基于 IPv4 架构的，随着计算机接入数量的不断增加，IP 地址资源愈加匮乏。为了解决该问题，各企业、学校及 Internet 服务提供商部署了大量 NAT，就 P2P 网络而言，它是构建在 Internet 网络之上的网络，Internet 上大量 NAT 设备的存在成为 P2P 应用普及和发展的障碍。

如上文所述，利用 NAT 技术，可以使一个局域网内的所有主机通过一个或几个公网 IP 地址来访问 Internet。但由于局域网与 Internet 编址方式的不同，NAT 设备掩藏了参与构建 PZP 网络的大量用户结点。不同的 NAT 后的计算机结点很可能是同一内部网络的 IP 地址，这样两个处在 NAT 后的计算机结点，无法利用高速的内部网络找到对方的地址并传

送数据，而只能和有公网 IP 地址的计算机通信。也只有获得了公网地址的计算机，才有可能与处于不同 NAT 后的计算机结点做转发数据工作，这样，对 P2P 网络而言，位于不同 NAT 之后的用户结点如何发现对方、如何彼此建立直接连接就成为 PZP 亟待解决的问题之一。

目前，很多在内部网中的计算机是通过 NAT 转换后的映射地址访问网络上的资源。这也是多数 P2P 软件穿越 NAT 的基本方法。下面就讲一下 P2P 网络中进行 NAT 穿透的相关方法。

5.4.1　UDP 概述

UDP 的全称是 User Datagram Protocol，是用户数据报协议。它是 OSI 参考模型中一种无连接的传输层协议，提供面向事务的简单不可靠信息传送服务，是一个简单的面向数据报的传输层协议，IETF RFC 768 是 UDP 的正式规范。

UDP 协议基本上是 IP 协议与上层协议的接口。UDP 协议适用端口分别运行在同一台设备上的多个应用程序。

由于大多数网络应用程序都在同一台机器上运行，计算机上必须能够确保目的地机器上的软件程序能从源地址机器处获得数据包，同时，源计算机能收到正确的回复。这是通过使用 UDP 的“端口号”完成的。

与 TCP 不同，UDP 并不提供对 IP 协议的可靠机制、流控制以及错误恢复功能等。由于 UDP 比较简单，UDP 头包含很少的字节，比 TCP 负载消耗少。

UDP 适用于不需要 TCP 可靠机制的情形，比如，当高层协议或应用程序提供错误和流控制功能的时候。UDP 是传输层协议，服务于很多知名应用层协议，包括网络文件系统（NFS）、简单网络管理协议（SNMP）、域名系统（DNS）以及简单文件传输系统（TFTP）、动态主机配置协议（DHCP）、路由信息协议（RIP）等。所以，在基于 P2P 技术的应用系统中，大部分的传输都是用 UDP 完成的。关于 UDP 的详细介绍可以参照相关文章。

5.4.2　UDP 穿透之中继（Relaying）

出现 NAT 设备后，中继是 peer-to-peer 通信最可靠的方法，但是却缺乏效率。NAT 设备通过中继进行通信，使 peer-to-peer 通信像 client/server 模式的网络。

在通信过程中，两个客户端主机 A 和 B，已经与有永久性 IP 地址的服务器 S 建立了一个初始连接。这两个客户端主机在不同的私有网络中，它们各自的 NAT 设备，阻止它们进行直接的连接。两个客户端不是试图进行直接的连接，而是简单地通过服务器 S 转发它们之间的消息。

这种方法的优点是只要两个客户端已经与服务器 S 建立了连接，它就可以一直工作。缺点是消耗服务器的处理能力和网络带宽，而且即使客户端与服务器建立了很稳定的连接，客户端之间的通信响应时间也可能会增加。

5.4.3　UDP 穿透之反向连接

一些 P2P 的应用程序采用了直接但是有所限制的技术来实现 NAT 穿越，该技术叫做

“反向连接”，这是用于当两个结点联入服务器 S 的时候，只有其中的一个结点在 NAT 设备的后面，如图 5.8 所示。如果 A 希望建立与 B 的连接，那么 A 可以直接联入 B，因为 B 是在公网中存在的，没有经过 NAT 转换，而且 A 的 NAT 设备也允许 A 直接由内网发起向外网的连接。如果 B 希望建立与 A 的连接，那么 A 的 NAT 设备就会阻止该操作，此时，B 可以借助于转发服务器 S，向 A 发送“反向连接”请求，由 A“主动”连接 B，从而达到 A 与 B 的 P2P 通信的目的。

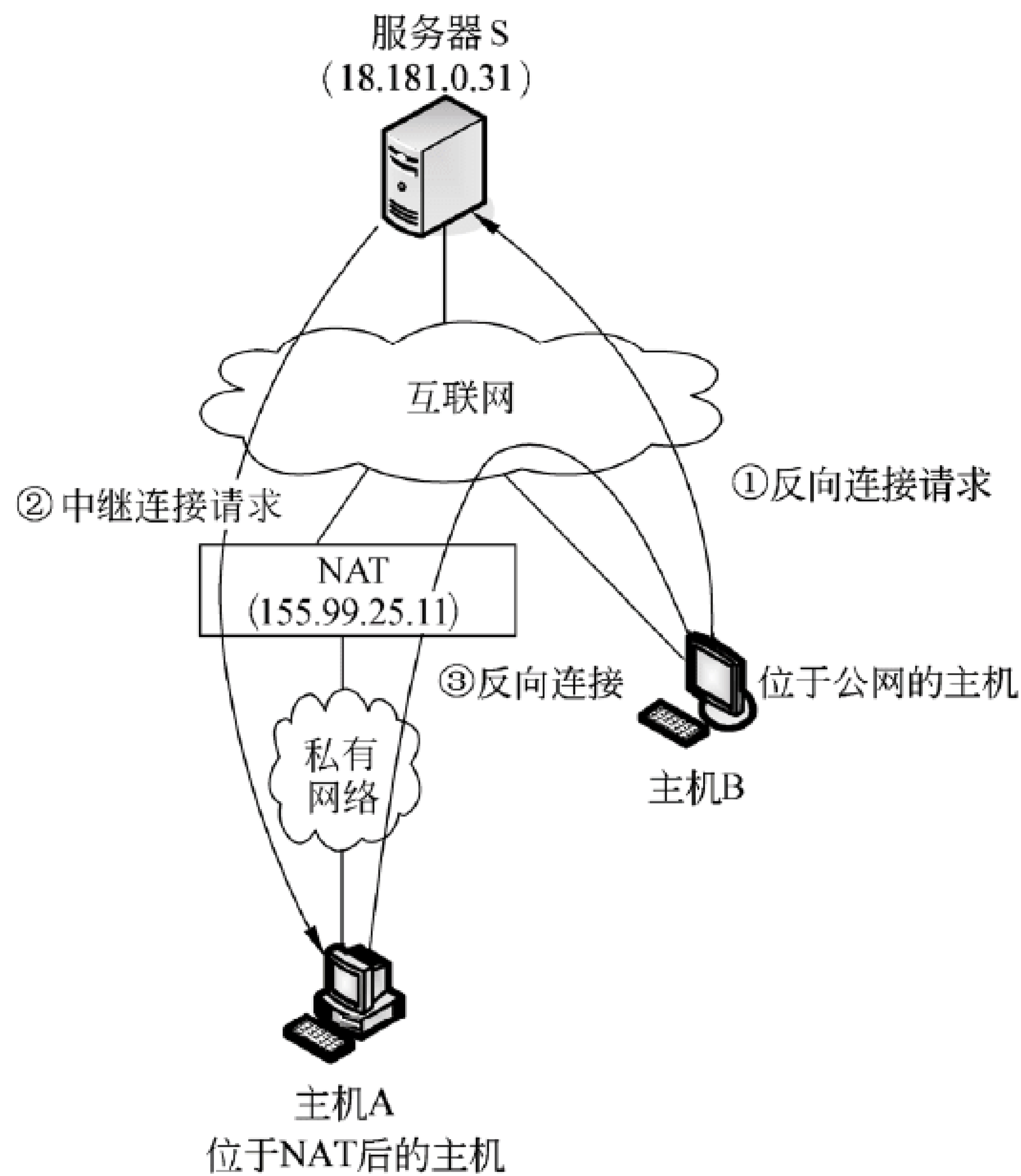


图 5.8　P2P 客户端反向连接穿透 NAT 示意图

尽管该技术的局限性非常明显，但是使用已知的服务器作为中介辅助 P2P 客户端双方进行 P2P 连接的思想已经成为了更加通用的“打洞”技术的基本思想。

5.4.4　UDP 穿透之 Holing 技术（UDP 打洞技术）

UDP 打洞技术依赖于防火墙和 Clone NAT 的属性、“洞”也就是所说的建立一个 Session（会话），通过恰当的设计 peer-to-peer 应用程序，可以使“洞”穿过 NAT 设备并在主机间可以建立连接。即使两个 P2P 客户端都位于 NAT 设备后面，UDP 打洞方式也能够通过已知的服务器实现 P2P 客户端直连。该技术可以参考 RFC 027 的第 5.1 节中的描述。

针对 P2P 客户端在网络中相对位置，可以有 3 种不同的情况。

1．P2P主机在不同的NAT后面

假定客户端 A 和 B 都有自己的私有 IP 地址，而且在不同的 NAT 设备后面，如图 5.9 所示。

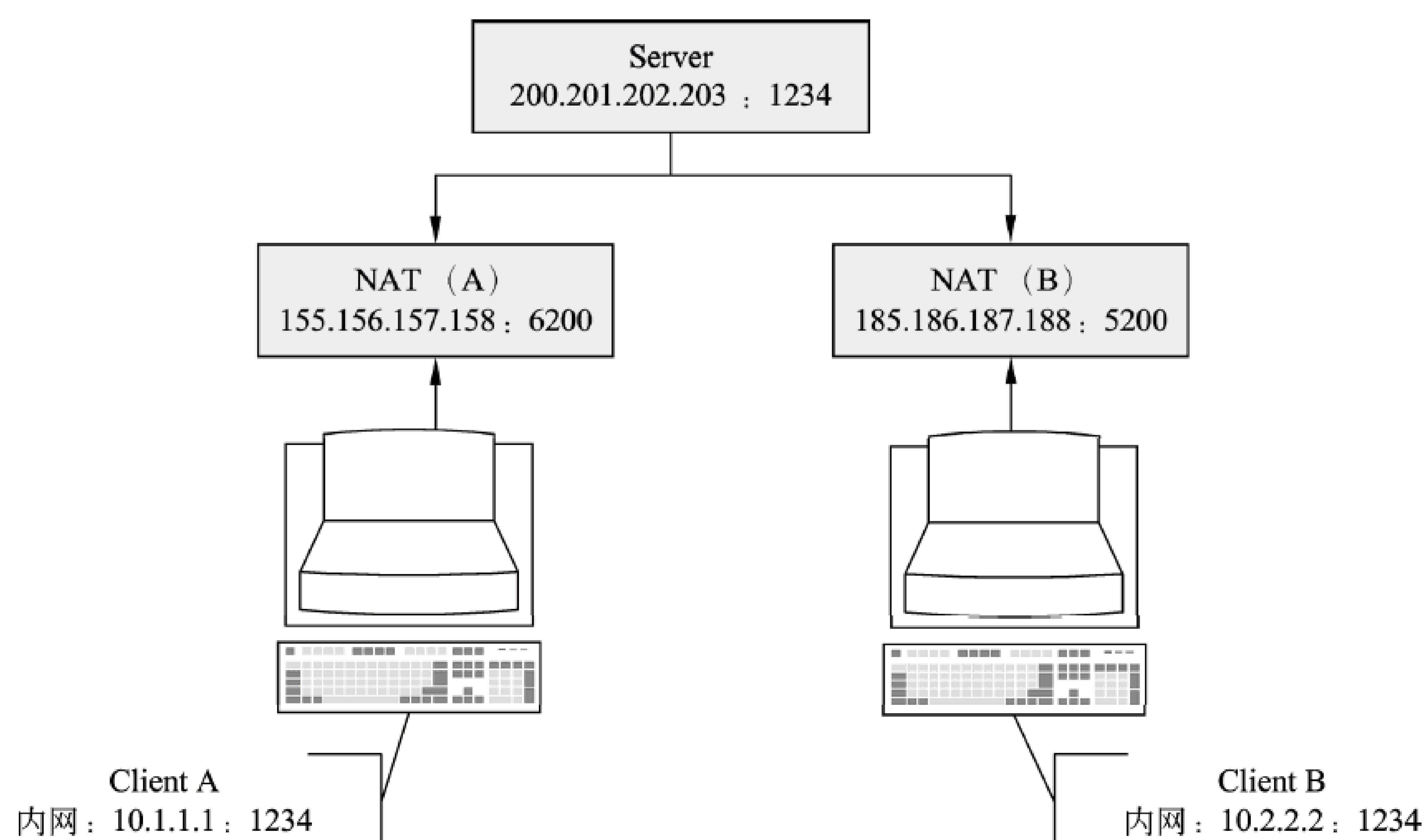


图 5.9　P2P 主机在不同的 NAT 后面的穿透过程

在客户端 A、B 和服务器 S 上都运行同一个 peer-to-peer 应用程序（如 P2P 共享下载，即时通信软件等），而且每一个主机都用 UDP 的端口号 1234。A 和 B 每一个都与服务器 S 初始一个 UDP 通信会话，这样就会使 NAT（A）为 A 与 S 的会话分配一个它自己的外网 UDP 端口 6200，同时 NAT（B）也为 B 与 S 的会话分配一个它的外网 UDP 端口 5200。

现在假设客户端 A 想与客户端 B 直接建立一个 UDP 通信会话。如果 A 只是简单地对 B 的外网地址（185.186.187.188：5200）发送一个 UDP 数据包，那么 NAT（B）很显然会丢弃这个发送过来的数据包（除非它是一个 Full Clone NAT 类型的 NAT）。因为 A 发送的数据包的源地址和端口号与 S 的不一致（开始时，NAT（B）与 S 建立了一个初始的会话，S 发给 NAT（B）的数据包的源地址与端口号为 200.201.202.203：1234）。同样地，如果 B 开始时向 NAT（A）发送一个 UDP 消息，那么 NAT（A）很显然也将丢弃这些信息。

假如 A 向 B 的外网 IP 发送 UDP 消息的同时，通过服务器 S 中继一个请求给 B，即请求 B 向 A 的外网地址发送 UDP 消息。A 向外的消息直接发到 B 的外网地址（185.186.187.188：5200）就会使 NAT（A）在 A 的自由地址与 B 的外网地址间建立一个新的通信会话。同时，B 发往 A 的外网地址（155.156.157.158：6200）消息会使 NAT（B）在 B 的私有地址与 A 的外网地址间建立一个新的会话。一旦在每一个方向都打开了新的 UDP 会话，那么客户端 A 与 B 可以直接进行通信而不会继续通过“中介”服务器 S 进行通信。

UDP 打洞技术有几个有用的属性，一旦在 NAT 后面的两个客户端建立了一个直接的 peer-to-peer 连接，那么在连接上的每一个结点都可以替代服务器 S 的“中介”的作用，帮助其他的结点与对等的客户端建立 peer-to-peer 通信，从而减轻服务器 S 的负担。如果采用 STUN，这种应用软件不需要探测它处在什么类型 NAT 后面。上面的通信过程表明：即使两个客户端的一个或两个都不在一个 NAT 设备后面，也能很好地建立 peer-to-peer 连接。即使有多个 NAT，UDP 打洞技术也能自动工作，甚至客户端在两层或更多层地址转换设

备后面，它仍然能被应用。

2．P2P主机在相同的NAT后面

当两个客户端恰好在相同的 NAT 后面，那么它们就在相同的 IP 地址空间里，如图 5.10 所示。

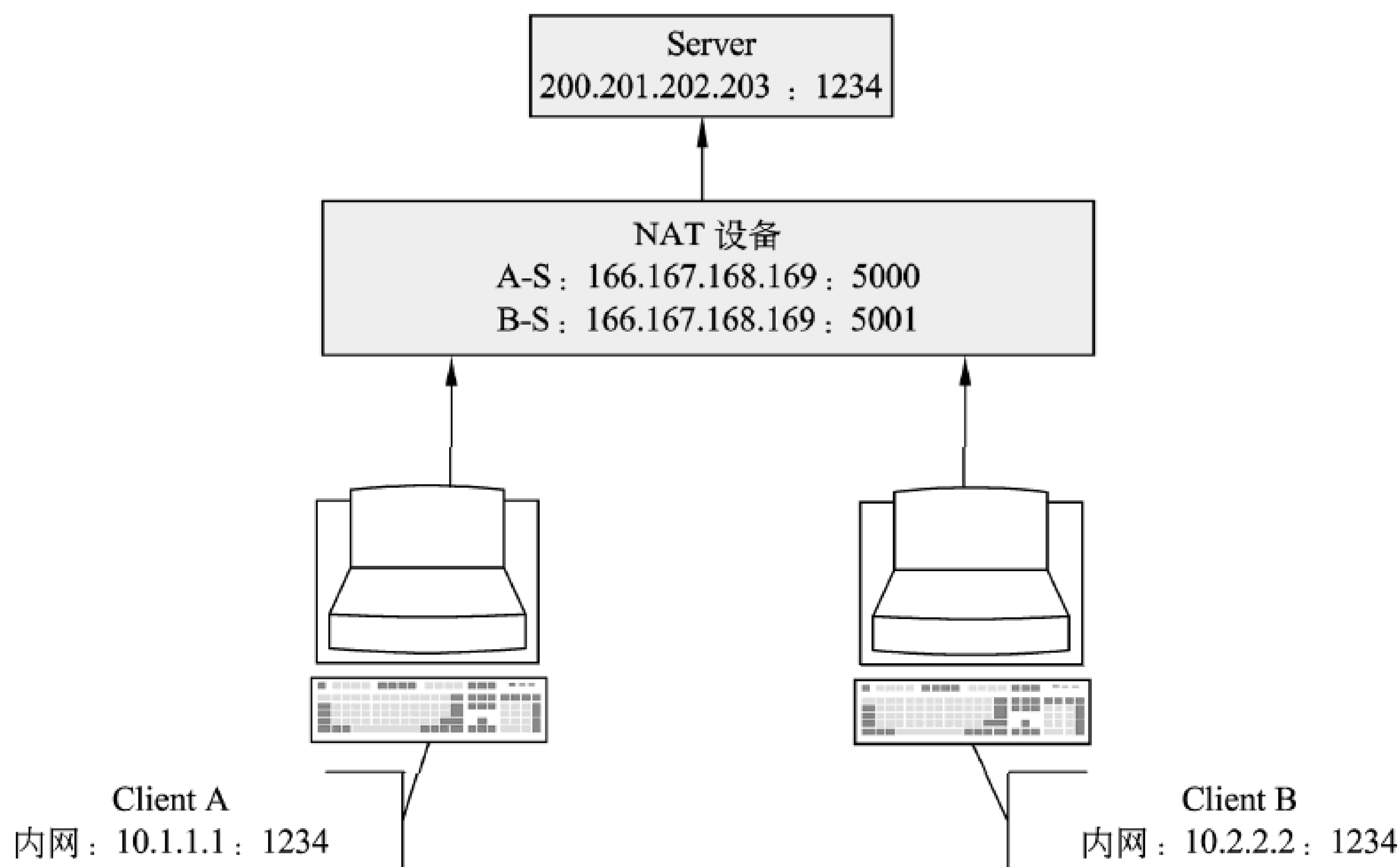


图 5.10　P2P 主机在相同的 NAT 后面的穿透情况

客户端 A 与服务器 S 建立了一个 UDP 会话，并且由 NAT 分配给一个外网端口 5000。同样地，客户端 B 也与服务器建立了一个会话，对应的端口号为 50001。

如果客户机 A 和 B 用服务器 S 作为中介，用上面介绍的 UDP 打洞技术建立一个通信连接。那么 A 和 B 得知道它们的公共 IP 地址和端口，如同服务器 S 看到的那样。然后 A 和 B 就在这些外网 IP 地址和端口上发送信息。只要这个 NAT 允许内网的主机可以与内网的其他主机打开转发 UDP 会话，而不能仅仅局限于与外部主机建立会话，那么这两个客户机可以用这种方式进行通信。

以上的这种情况称为“回环翻译”（loop back translation），因为当从内网的数据包到达 NAT 后，就被翻译，再被“回环翻译”到内网中，而不是要穿过 NAT 经过公网（public network）。例如当 A 向 B 的外网地址发送一个 UDP 数据，这个数据包初始的时候有一个源 IP 地址和端口号（10.1.1.1：1234）和一个目的地址与端口（166.167.168.169：5003）。

NAT 收到这个数据包，把它翻译为 166.167.168.169：5000（A 的公共地址）的源地址和 10.2.2.2：1234 的目的地址，然后就把它转发给 B。即使 NAT 支持“回环翻译”，在这种情况下翻译与转发是不必要的步骤，很可能在 A 和 B 之间加上一个潜在的对话，同时也加重了 NAT 的负担。这种情况下，可以不通过 NAT 直接向它们的私有地址发送数据包。

3．P2P对等的主机由多个NAT分开

在有多个 NAT 设备的拓扑结构中，如果不知道拓扑结构的细节，将很难在两个客户端间建立一个最优的 P2P 路线，例如图 5.11 所示的情况。

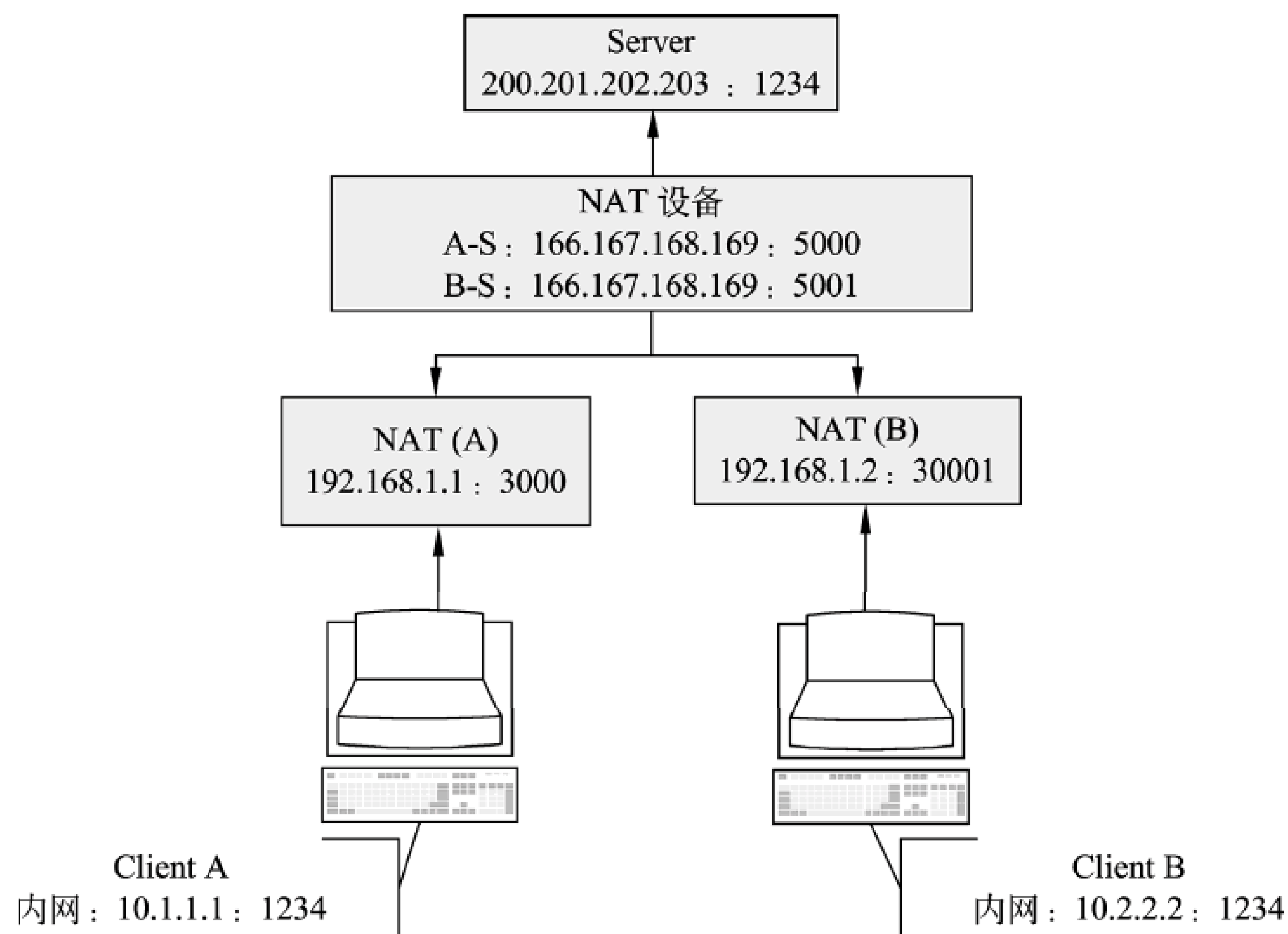


图 5.11　P2P 主机由多个 NAT 分开的结构图

假定 NAT（X）是一个因特网服务供应商（ISP）配置的一个大的工业型 NAT。这个 ISP 可以通过很少的一些全球唯一的 IP 地址让众多的客户上网查询各种信息。NAT（A）和 NAT（B）是两个消费者为自己的私有家庭网络配置的 NAT。只有服务器 S 和 NAT（X）有全球唯一的 IP 地址，NAT（A）和 NAT（B）的 IP 地址是 ISP 的私有地址，客户机 A 与客户机 B 的 IP 地址又分别是 NAT（A）和 NAT（B）的私有地址。每一个客户机如前面所讲的那样，开始的时候向服务器 S 初始连接，这样可以使 NAT（A）和 NAT（B）创建一个 public/private 转换，同时也会使 NAT（X）为每一个会话建立一个 public/private 转换。

现在假如客户机 A 和 B 试图建立一个直接的 peer-to-peer UDP 连接。对客户机 A 最佳的方法就是直接向客户机 B 在 NAT（B）上的 IP 地址 192.168.1.2：3001 发送信息。同样客户机 B 可向客户机 A 在 NAT（A）上的 IP 地址 192.168.1.1：3000 发送消息。然而，A 和 B 不知道它们的这些 IP 地址，因为服务器 S 才能看到客户机“全局”（global）IP 地址，即 166.167.168.169：5000 和 166.167.168.169：50001。即使 A 和 B 通过别的方式知道这些 IP 地址，仍然不能保证它们是可用的，因为在 ISP 的私有地址分配中可能与客户机的私有地址的分配有冲突。因此，客户机不得不用它们全局的公共地址，如服务器 S 看到的地址那样，来进行它们的 P2P 通信，就得靠 NAT（X）提供回环翻译的功能。

5.4.5　UDP 穿透 NAT 的实例

下面以一个实例来说明在 UDP 的连接下进行 NAT 的穿透，如下就是 UDP 连接下穿透 NAT 的过程。

（1）A 登录 Server，NAT　A 分配端口 11000，Server 得到 A 的地址为 100.10.10.10：11000。

（2）B 登录 Server，NAT　B 分配端口 22000，Server 得到 B 的地址为 200.20.20.20：22000。

（3）此时 B 会把直接来自 A 的包丢弃，所以要在 NAT B 上打一个方向为 A 的洞，那么 A 就可以向 200.20.20.20：22000 发送数据了。

（4）打洞的指令来自 Server。B 向 A 的地址 100.10.10.10：11000 发一个 UDP 报文，被 NAT A 丢弃，但在 NAT B 上建立映射记录，NAT B 不在丢弃来自 A 的报文。

（5）Server 通知 A 可以通信，A 发起数据 UDP 包给 B，NAT B 放行，B 收到 A 的包，双方开始通信。

注意：若是对称 NAT，当 B 向 A 打洞的端口要重新分配（NAT A 不会再分配 11000 端口），B 无法获取这个端口，所以不适用本方法。

5.5　TCP 连接下的 NAT 穿透技术

相对于 UDP 连接方式下的穿透来说，TCP 连接下的穿透则麻烦得多。因为 TCP 是一种面向连接（连接导向）的、可靠的、基于字节流的运输层（Transport layer）通信协议，建立 TCP 连接需要 3 次握手（hand shaking）机制才能完成。在基于 P2P 的应用中，大部分都是基于 UDP 连接的，所以 TCP 连接下的 NAT 穿透在这里只作简要介绍，有兴趣的读者可自行研究。

5.5.1　TCP 简介

TCP 是一种面向连接（连接导向）的、可靠的、基于字节流的运输层（Transport layer）通信协议，由 IETF 的 RFC 793 说明（specified）。在简化的计算机网络 OSI 模型中，它完成第四层传输层所指定的功能，同一层内另一个重要的传输协议就是上面讲述的 UDP 协议。

在因特网协议族（Internet protocol suite）中，TCP 层是位于 IP 层之上，应用层之下的中间层。不同主机的应用层之间经常需要可靠的、像管道一样的连接，但是 IP 层不提供这样的流机制，而是提供不可靠的包交换。

应用层向 TCP 层发送用于网间传输的、用 8 位字节表示的数据流，然后 TCP 把数据流分割成适当长度的报文段。之后 TCP 把结果包传给 IP 层，由它来通过网络将包传送给接收端实体的 TCP 层。

注意：文中所说的“适当长度的报文段”，通常受该计算机连接的网络数据链路层 MTU 的限制。这里 MTU 指的是 Maximum Transmission Unit 的意思，即最大传输单元，是指一种通信协议的某一层上面所能通过的最大数据报大小，以字节为单位计算。最大传输单元这个参数通常与通信接口有关，如网络接口卡、串口等。

TCP 为了保证不发生丢包，就给每个字节一个序号，同时序号也保证了传送到接收端实体的包的按序接收。然后接收端实体对已成功收到的字节发回一个相应的确认（ACK）。

如果发送端实体在合理的往返时延（RTT）内未收到确认，那么对应的数据（假设丢

失了）将会被重传。TCP 用一个校验和函数来检验数据是否有错误；在发送和接收时都要计算校验和，传输过程有以下几个特点。

- TCP 建立连接之后，通信双方都同时可以进行数据的传输。
- 是全双工通信的。
- 在保证可靠性上，采用超时重传和捎带确认机制。
- 在流量控制上，采用滑动窗口协议，对于窗口内未经确认的分组需要重传。
- 在拥塞控制上，采用慢启动算法。

注意： 滑动窗口协议，是 TCP 在进行数据传输时使用的一种协议，通过一种滑动窗口的机制来暂存两台计算机间要传送的数据分组以解决传输效率和流量控制问题。此协议的基本原理是：在任意时刻，发送方都维持了一个连续的允许发送的帧的序号，称为发送窗口；同时，接收方也维持了一个连续的允许接收的帧的序号，称为接收窗口。发送窗口和接收窗口的序号的上下界不一定要一样，甚至大小也可以不同。不同的滑动窗口协议窗口大小一般不同。发送方窗口内的序列号代表了那些已经被发送，但是还没有被确认的帧，或者是那些可以被发送的帧。

总之 TCP 协议概括起来就是：面向连接的、端到端的、采用字节流方式的、高可靠的、全双工的传输方式。具有提供计算机程序间连接、检测和丢弃重复的分组、完成数据报的确认、流量控制和网络拥塞等多种功能，在文件传送（File Transfer）、远程登录（Remote login）、计算机邮件（Mail）、网络文件系统（NFS）、远程打印（Remote printing）、远程执行（Remote execution）、名字服务器（Name servers）、终端服务器（Terminal servers）等多方面都有重要的应用。

5.5.2　TCP 穿透 NAT 的方式

在 TCP 中用 3 次握手机制建立连接时，若对方的（hole punching）还没有准备好，则发送 SYN 包给对方时，视对方 NAT 处理方法而定，对方会回送一个 RST 封包或者直接把该包丢弃。在下次连接时，端口号已经发生了变化，则两方的连接不能成功。所以，必须由一方先发送一个 UDP 封包通知对方立刻传送 SYN 包，而己方则在 UDP 封包发出去后的一小段时间才发送 SYN 封包，才可以使得对方（hole punching）准备好。

这种方法相对依赖于网络的时延，比较少采用。因此，在 NAT 环境下，P2P 程序一般采用 UDP 的通信方式。下面就以一个最简单的实例来说明 TCP 连接下的 NAT 穿透。

TCP 连接下穿透 NAT：

- A 登录 Server，NAT A 分配端口 11000，Server 得到 A 的地址为 100.10.10.10：11000。
- B 登录 Server，NAT B 分配端口 22000，Server 得到 B 的地址为 200.20.20.20：22000。
- A 向 B 发送 TCP 数据包 SYN：192.168.10.11：1234=>200.20.20.20:22000，在 NAT A 上打洞。
- B 向 A 发送 TCP 数据包 SYN：192.168.20.22：1234=>100.10.10.10：11000，在 NAT B 上打洞。
- 通道建立，A 与 B 经过 3 次握手建立 TCP 连接。

5.6　NAT 穿透在 P2P 系统中的应用

在基于 P2P 的应用系统中，有很多系统都需要 Peer 之间的交互和通信，如 BT、eMule 和迅雷下载等，它们都是 P2P 技术在文件传输和共享领域的重要应用，也是当前互联网上非常流行的一种文件下载方式，但是它们都有一个致命的问题，就是其生命周期太短。当网上的种子数为 0 或 peer 数目变得更少时，其健康度有可能会小于 100%。在这种情况下，极有可能无法再下到相应的资源。出现这种情况，有时并不是真的没有“源”，而是很多拥有“源”的 Peer 都隐藏在 NAT 设备之后，不经过穿透是无法找到这些 Peer 的。本节就讲述一下在实际的 P2P 应用系统中是如何进行 NAT 穿透。

5.6.1　eMule 穿透内网的原理

eMule 是 P2P 技术的一个重要应用之一，建立于多点文件传输协议之上。一个电驴网络由服务器端和客户端两部分组成。客户端是用户与服务器建立连接、进行搜索下载并提供操作界面的应用系统，服务器端，就是分布在全球的、可数的 eMule 服务器，客户通过浏览 eMule 服务器而获取他需要的文件信息以及拥有此文件的 Peer 信息。eMule 下载过程完全是客户端之间的交互，不通过服务器端。

eMule 本身作为一个基于 P2P 的应用系统，主要提供搜索和下载两大类功能。具体的 eMule 详细情况，请参阅本书第 6 章所讲的内容。这里只对 eMule 中的穿透方法及原理进行简要说明。

1．eMule中的高ID和低ID

高 ID 在 eMule 协议中就是 HIGH ID 的意思，指的是拥有独立外网 IP 地址并且能把端口开放给 eMule 的用户。

eMule 默认的开放端口是 4662，当然这个端口是可以更改的。此类用户可以和任何高 ID 或是低 ID 电驴客户端连接并上传下载数据。简单地说，如果 eMule 网络中的一个 Peer，它拥有独立外网 IP 地址并且能把端口开放出来的话，那么 eMule 服务器就会分给这个 Peer 一个 High ID

- 低 ID 就是 LOW ID 的意思，是与 High ID 相对应的，指的是那些没有外网 IP 地址的内网 eMule 用户。简单地说，在 eMule 网络中，那些在防火墙后面没有公网 IP 地址的 Peer 在接入到 eMule 网络后，就被系统自动分配为低 ID。
- 高 ID 可以与低 ID 连接并通信，低 ID 可以与高 ID 连接并通信，但是两个低 ID 用户之间是无法直接连接的，所以低 ID 只能从高 ID 那里得到资源，而无法从同样是低 ID 的用户那里下载资源。

eMule 网络中的高 ID 与低 ID 之分，主要还是由于内外网的原因。Internet 网络协议造成的两个内网用户是无法直接通信的，而是要经过服务器中转才行。一般的通信软件如 QQ、MSN 如果是不经过穿透而在内网通信的话，就要经过中转才能到达对方那里。但是由于 EMULE 传输的资源超大，所以不可能有大型服务器去做 eMule 的中转，这样就需要在两个低 ID 之间架起一座桥梁，也就是 low2low 技术。而这种技术就是 eMule 中的穿透

技术，也就是实现两个 LowID 之间的通信。

2．eMule中的穿透

内网计算机，也就是上面所说的 LowID 用户，都通过至少一层网关连接互联网，没有自己的独立 IP 和端口，所以其他的 Peer 无法主动与你建立连接，两个内网用户自然也就无法连通，更无法实现传输。

但是内网计算机可以主动连接其他有独立 IP 的外网计算机，也就是 High ID 用户。在通过 UDP 协议通信的时候，因为 UDP 是非持续连接的，所以网关会开放一个临时端口。LowID 的用户能够接收外网计算机返回的 UDP 包，如果一段时间内没有传输，临时端口便会取消。

在这个过程中，就可以加入穿透机制了。比如 A 和 B 两台内网计算机，都同时连接外网计算机 C 进行 UDP 协议的传输，A 和 B 分别用到了临时端口 Ap 和 Bp，这个时候通过 Ap 就可以主动连接到 A，Bp 就可以主动连接到 B，所以 C 所要做的，就是把 Ap 告诉 B，把 Bp 告诉 A。AB 通过从 C 那里知道的 Ap 和 Bp，即可实现 UDP 直连。

只要连接不断，临时端口就一直有效，传输期间，C 什么都不需要参与，这个过程，就是上文所讲的 UDP 打洞技术，C 帮 AB 打好洞，AB 就可以自己玩了。

当然这是 eMule 中 NAT 穿透最简单的描述，根据不同的网关设备，还有很多不同的问题需要解决。

5.6.2　使用 BT 下载时 NAT 的穿透设置

BT 下载是 P2P 技术的重要应用之一，也是目前最热门的下载方式之一，中文全称为“比特流”，在网络中也有人称之为“变态下载”。从本质上讲。BT 只是基于 P2P 技术的一种资源共享和传输的应用系统。

1．BT是怎样下载文件的

就传统的 HTTP、FTP、PUB 等下载方式而言，它们都是基于 C/S 模式的，也就是首先将用于发布的资源放到服务器上，然后再由服务器传送到每位用户的机器上，它的工作原理如图 5.12 所示。

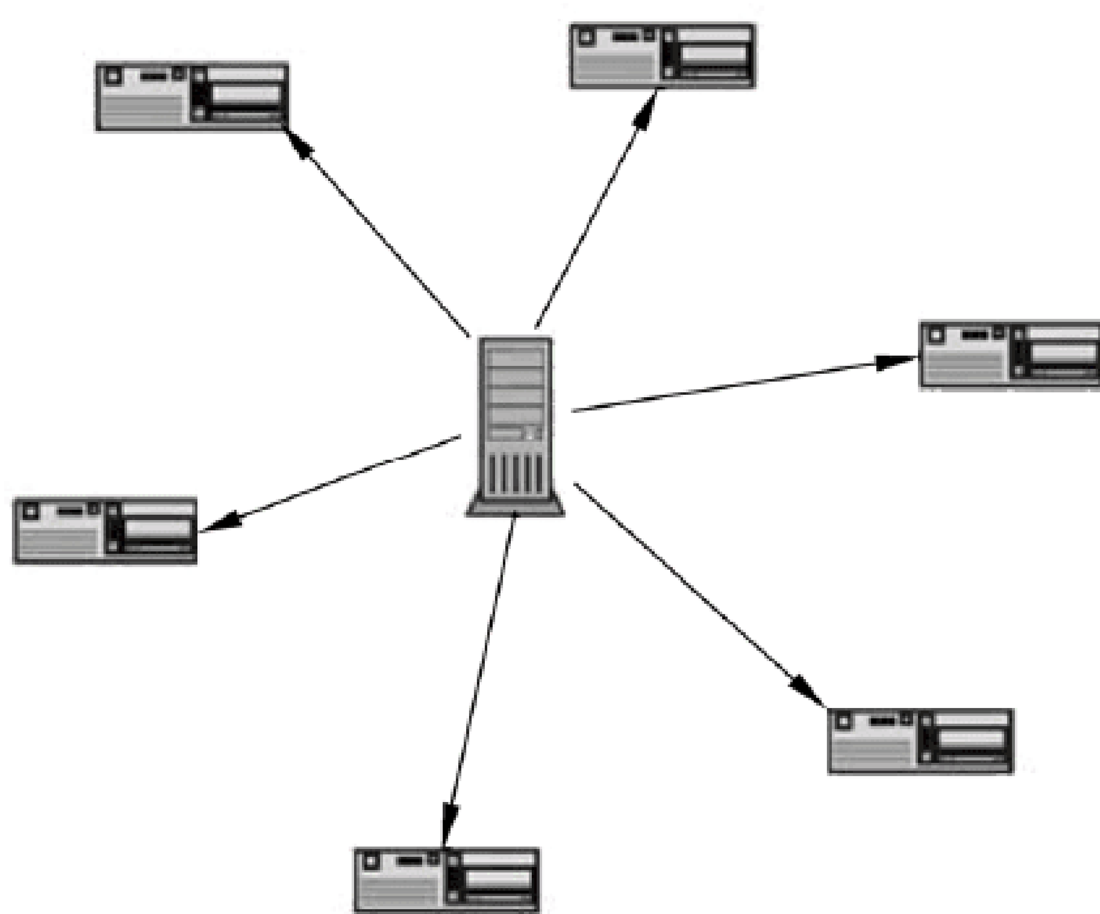


图 5.12　基于 C/S 模式的传统下载方式

由图 5.12 所示的下载模式中，服务器负责应答来自所有客户端的请求。如果同一时刻下载的用户数量太多，势必让服务器承受过多的负载，影响所有用户的带宽和下载速度都有影响，如果用户超过一定的限制，服务器甚至会发生崩溃。

而 BT 下载则是基于 P2P 技术实现的，整个下载过程不经过服务器，直接在 Peer 之间完成，它的下载原理如图 5.13 所示。

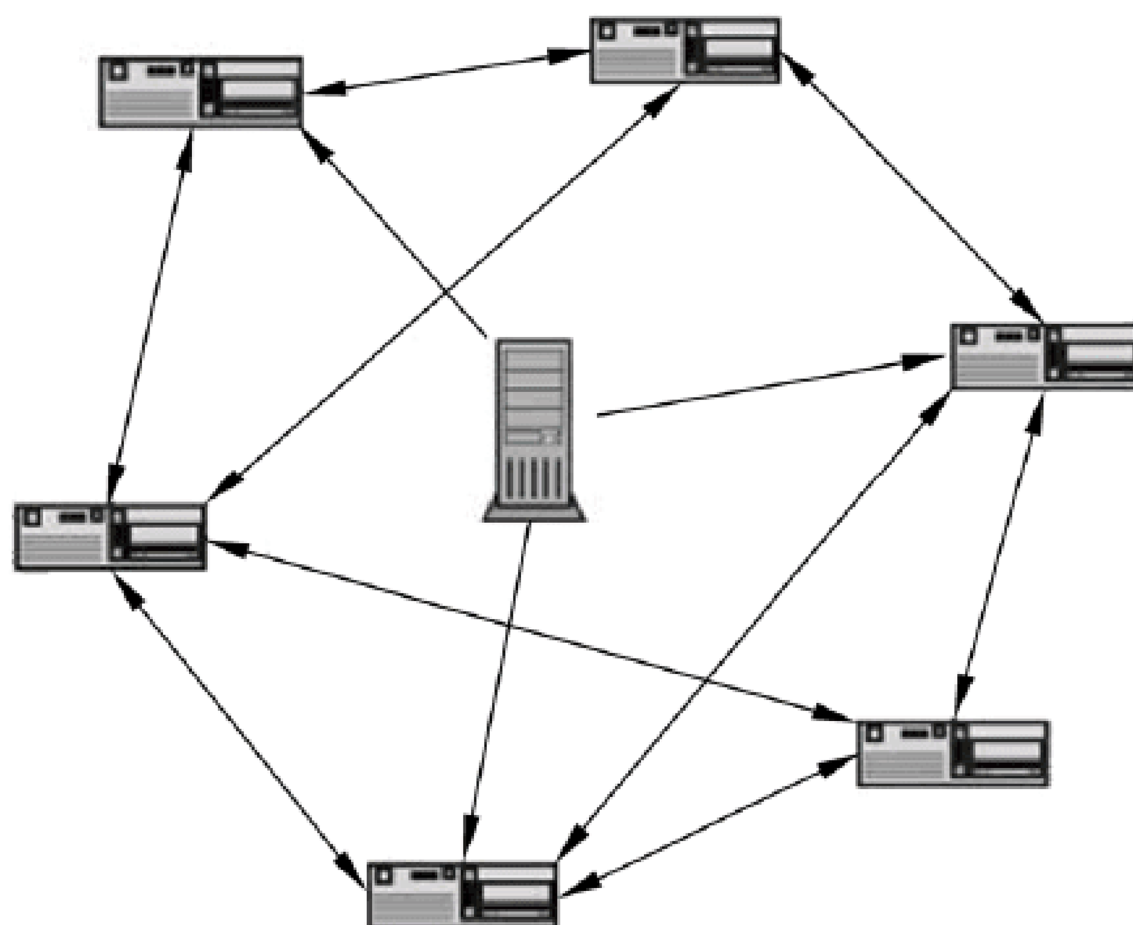


图 5.13　基于 P2P 方式的下载原理

从图 5.13 就可以清楚地看出，每台参与下载的计算机既从其他用户的计算机上下载文件，同时自身也向其他用户提供下载。下载过程是通过 Peer 之间的交互来完成的，因此参与下载的用户数量越多，下载速度也越高。关于 BT 下载的详细情况，请参阅本书第 6 章所讲的内容。

2．为什么需要穿透

上文已经对 BT 下载的基本原理和方法进行简要的说明，也就是说，BT 下载最关键的问题就是要找到可以给你提供文件下载的 Peer。在 P2P 搜索技术中已经说过，要定位某一个 Peer 只需知道它的 IP 地址和端口就可以了，如果是两个存在于公网中的 Peer，可以轻松地实现交互和 Peer 间的通信。但是，如果是内网的 Peer 与一个公网的 Peer 进行通信，或者是两个内网的 Peer 进行通信，就需要进行穿透了。

就目前我国情况来说，公网 IP 地址偏少，绝大部分用户的上网方式都是通过 NAT 技术进行共享上网的，加入到 BT 网络中的很多 Peer 也都隐藏在内网之中。在内网下，公网的主机看不到内网主机的 IP 地址，只能看到内网主机通行的网关。如果对方 Peer 也在内网下，那么这两个 Peer 之间就不能互传文件，而只有通过穿透技术才能实现内网 Peer 之间的互联和通信。

关于 BT 下载的穿透技术，有多种方法可以实现。当前众多流行的 BT 下载客户端，大部分都实现了对 NAT 的穿透，只需开放相应通信端口即可。这里只对常见的 BT 下载时进行内网穿透的基本设置方法进行说明。

3．BT穿透的设置方法。

其实，对 BT 进行 NAT 穿透的设置，就是一个针对防火墙的设置，一旦内容中的用户主机上安装了防火墙，那么防火墙阻挡来自外网的连接，别的 Peer 不能主动地连接到你的

机器上，BT 的下载速度当然也就不会快起来的。在安装有防火墙的机器要想要 BT 下载速度加快的话，就要使 BT 下载软件穿透防火墙的阻隔，让其能够连接到外网的 Peer。下面就针对两种最常用的防火墙设置进行说明。

BT 下载有很多客户端可以选择，大部分客户端都支持针对防火墙的穿透设置，而且设置方法也都大同小异。本文以 BitTorrent Plus 这个客户端为例来说明其穿透 Windows 防火墙和瑞星防火墙的设置。

BitTorrent Plus 使用的默认下载端口是 6881～6889，当然，该软件也允许用户自行指定 BT 的下载端口、对 BT 的穿透设置。其他就是设置相应的防火墙，对 6881～6889 的端口放行，不拦截 BT 通过 6881～6889 进行通信的所有数据包。

4．Windows防火墙的设置方法

XP 自带有 ICF，是 Internet Connection Firewall 的简称，也就是因特网连接防火墙。ICF 建立在我们的电脑与因特网之间，它可以让你请求的数据通过，而阻碍你没有请求的数据包，它是一个基于包的防火墙。在使用 BT 时会因为 ICF 的阻拦，引起连接不到 Peer 或者数据包延滞降低下载速度，所以需要在 ICF 中设置，不阻拦 BT 使用的端口。

具体方法如下。

（1）在系统桌面上用鼠标左键单击“开始”按钮，在弹出的菜单列表中，选择“设置”|“控制面板”命令。在控制面板的选项列表中，双击打开“网络连接”。

（2）在出现“网络连接”的对话框中，用右键选择本机上网所用的网络连接，在弹出的菜单列表中选择“属性”选项。

（3）接着会弹出一个“本地连接属性”对话框，选择其中的“高级”标签并单击对话框下方的“设置”按钮。

（4）在弹出的“Windows 防火墙”对话框中，再次选择其中的“高级”标签。在“网络连接设置选项下”单击“设置”按钮。

（5）在出现的“高级设置”对话框中选择其中的“服务”标签并单击“添加”按钮，弹出“服务设置”对话框，如图 5.14 所示。

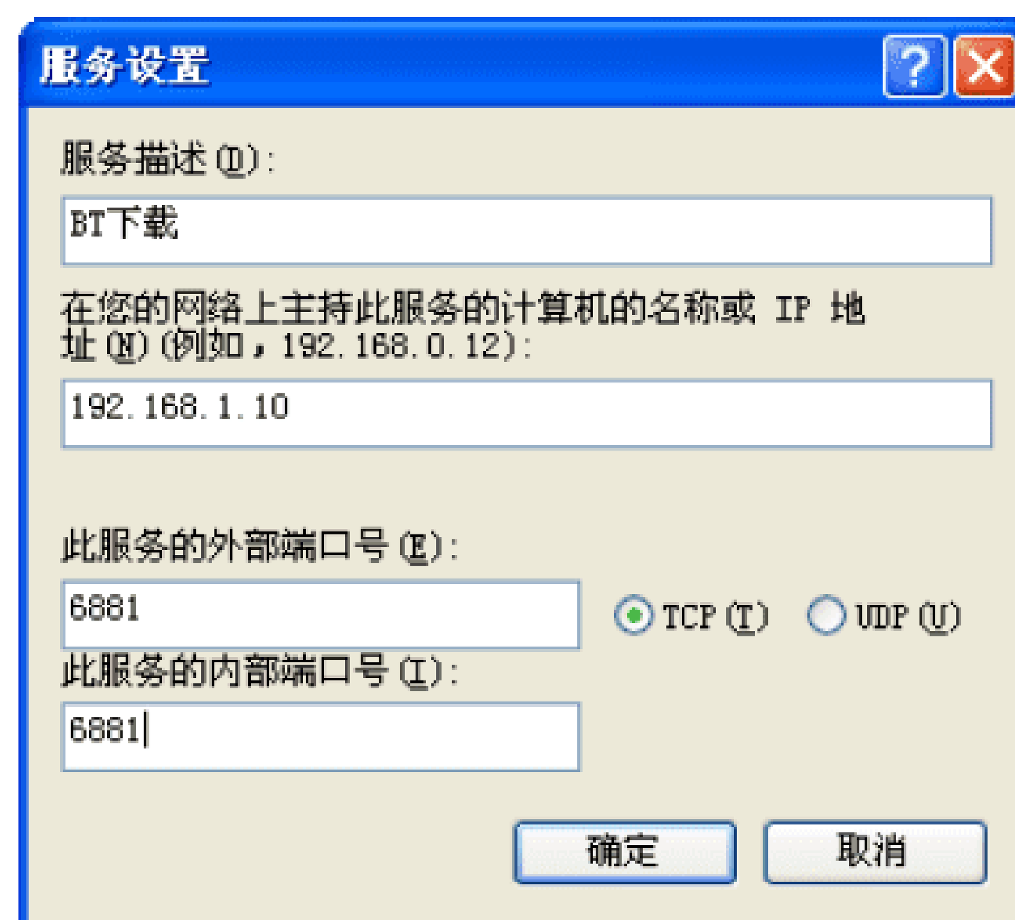


图 5.14　服务设置的界面图

（6）在出现的“服务设置”对话框中，其中的“服务描述”文本框中是对该网络服务的描述，可填入“BT 下载”字样。“在您的网络上主持此服务的计算机的名称或 IP 地址”文本框中是要填入要进行 BT 下载机器的 IP 地址或电脑名称，即你的电脑名称。在“此服务器的外部端口号”文本框中填写 6881，在“此服务器的外部端口号”文本框中也填写 6881。选中 TCP 协议然后单击“确定”按钮即可。这样设置就可以开放 6881 端口的服务了，按同样的设置方法可以开放 6882～6889 端口的服务。操作完成后即可解决问题。

5．瑞星防火墙的设置方法

瑞星防火墙是当前 Windows 用户中使用的一款较为普遍的防火墙系统，在瑞星防火墙

中设置 BT 的穿透也很容易，就是添加一条防火墙过滤规则。具体的设置方法如下。

（1）打开瑞星防火墙的主界面，选择“设置 | 详细设置”菜单，进入瑞星防火墙的详细设置主界面，如图 5.15 所示。



图 5.15　瑞星防火墙设置的主界面图

（2）在图 5.15 所示的界面中，选择左侧树型菜单中的“规则设置 | IP 规则”选项，然后在显示 IP 规则的界面下，单击“增加”按钮，弹出“IP 规则设置”对话框，如图 5.16 所示。

（3）在图 5.16 所示的 IP 规则设置中，填入规则名称，如“BT 规则”。设置规则应用于所有的 IP 包，对触发本规则的 IP 包设置放行，然后单击“下一步”按钮。

（4）在接下来的设置中，在“设置通信的本地电脑地址”中，指定本机 IP 地址即可。在“设置通信的运程电脑地址”中，选择任意地址，再单击“下一步”按钮。

（5）接着，会弹出如图 5.17 所示的对话框。在这项设置中，在“协议”文本框中选择 TCP，在“对方端口”文本框中选择端口范围。起始端口设置为 6881，结束端口设置为 6889，设置后的效果如图 5.17 所示



图 5.16　瑞星防火墙的 IP 规则设置



图 5.17　瑞星防火墙的端口设置界面图

（6）直接单击“下一步”按钮，完成对瑞星防火墙的设置。

注意： 在改变规则设置或添加了新规则后要重新开启防火墙，以使最新的防火墙规则生效。而且，不同的防火墙版本，其具体的设置方式也有所不同，只要保证防火墙对 BT 对外连接的端口开放即可。

针对其他的防火墙，也有不同的设置方法，这里就不再详说。只要保证在 BT 客户端下载的时候，开启其下载的通信端口，基本上就能实现 NAT 的穿透，实现两个内网 Peer 之间的通信。

5.6.3　迅雷下载的穿透技术

前面对 eMule 的穿透原理和 BT 的穿透设置已做了说明，下面要说的是迅雷下载的穿透技术。迅雷，也是一个用于下载的软件，其本身不支持上传资源，它只是一个提供下载的工具软件，可以支持多种协议下载，如 HTTP、FTP、MMS、RTSP、BT、EMULE 等。既然迅雷可以下载 BT、eMule，那么也必定存在同样的穿透问题，也就是说，迅雷是如何找到更多的 Peer，如何与 NAT 设备后面的 Peer 进行通信的。

要想提升迅雷的下载速度，同时还要下载到完整资源的关键，在于所连接的 peers 中的数据部分能否构成一个完整资源，这样就要求 P2P 的下载工具能够连接上更多的 peers。

迅雷下载技术支持 UPnP 端口自动映射技术，这意味着，我们可以接受外网 peers 的主动连接，这样就可以连上更多的 peers。加上了更多的 peers，其构成一个完整资源的概率将大大提升，从而可以保证“文件可以下全”。同时，我们不仅可以从这些 peers 处获得数据，还可以将自己下载的部分上传给其他人。而根据 P2P 精神，人人为我，我为人人，上传越多，下载速度就会越快。这样，就进入了良性循环。

5.7　UPnP 的 P2P 端口映射技术

在上文的 5.6 节中，提到了一种 UPnP 的端口自动映射技术，UPnP 是各种各样的智能设备、无线设备和个人电脑等实现遍布全球的对等网络——P2P 连接的结构。在 P2P 网络中应用 UPnP 技术，意味着对于一台内网电脑（即只具有内部 IP 地址的电脑，多数企业局域网用户和使用宽带路由器用户均在此类范围），P2P 网络中的 UPnP 功能可以使网关或路由器的 NAT 模块做自动端口映射，使基于 P2P 的应用系统将监听的端口从网关或路由器映射到内网电脑上。这样一来，就可以更好地突破内网限制，增强穿透能力更强，也能够连接到更多的隐藏在 NAT 设备之后的 Peer，P2P 网络的连通性和交互性也就更好。本节就重点讲解一下 UPnP 的相关知识及其穿透 NAT 的功能。

5.7.1　UPnP 的概念

UPnP 的全称是 Universal Plug and Play，意思是通用即插即用，是一种用于 PC 和智能设备（或仪器）的常见对等网络连接的体系结构。它是一种分布式的，开放的网络架构，UPnP 可以充分发挥 TCP/IP 和网络技术的功能，不但能对类似网络进行无缝连接，而且还

能够控制网络设备及在它们之间传输信息。UPnP 是独立的媒介，在任何操作系统中，利用任何编程语言都可以使用 UPnP 设备。

UPnP 的这种特性使得其应用范围非常大，包括家庭自动化、打印、图片处理、音频/视频娱乐、厨房设备、汽车网络和公共集会场所的类似网络，尤其是在家庭中。UPnP 以 Internet 标准和技术（例如 TCP/IP、HTTP 和 XML）为基础，使这样的设备彼此可自动连接和协同工作，从而使网络化（尤其是家庭网络）对更多的人成为可能。

5.7.2　哪些用户需要用 UPnP 功能

UPnP 的应用是有目的性的，并不是所有的网络活动都需要 UPnP 的支持，一般以下几种情况的用户需要使用 UPnP 的功能。

- 只有在需要使用一些支持 UPnP 功能的 P2P 软件时，如 BT、电骡 eMule、MSN 等，我们才需要考虑 UPnP。如果你根本就不用这些软件，仅仅是上网浏览的话，也就没必要使用 UPnP 的功能了。
- 如果需要使用这些 P2P 软件，但是外网用户时，也无须设置 UPnP，因为你不需要做什么 UPnP 就可以正常使用这些 P2P 软件了。
- 如果你是内网用户，但已经手动为这些 P2P 软件进行了端口映射，如在使用 BitComet（比特慧星）下载时，在“用户列表”中已经看到“远程”，或者是使用电驴 eMule 连接服务器成功后，已经显示为高 ID，那么 UPnP 也用不着了。

注意： 手动做的端口映射只是针对某个 P2P 软件起作用，如果再使用新的 P2P 软件，仍然需要针对新的 P2P 软件做相应的端口映射才可以。

- 如果你是内网用户，需要使用这些 P2P 软件，而且并未进行手动端口映射。比如在使用 BitComet 进行下载时，“用户列表”中只有“本地”而没有“远程”，在使用电驴 eMule 时，显示的也是低 ID，那么此时才需要考虑端口映射的问题。

通过以上所说的 4 点，结合上文讲述的 NAT 穿透，不难看出，在 P2P 网络中使用 UPnP 的功能，简单地说就是做 NAT 穿透的，就是通过 UPnP 的端口映射功能实现外网 Peer 与内网 Peer 的互联和通信。

5.7.3　UPnP 在 P2P 网络中的应用

UPnP 作为一种通用的即插即用网络设备，可以充分发挥 TCP/IP 和网络技术的功能，能够控制网络设备及在它们之间传输信息，还可以进行端口映射。在 P2P 网络中 UPnP 有着重要的应用，常见的 UPnP 在 P2P 网络中的应用主要有以下两点。

1. 网络地址转换

在数量以百万计算而且数目仍然在继续增长的家庭网络出现以前，Internet 上的寻址系统就已经开发出来了。实际上，在 Internet 尚处于幼年的时候所开发的这个寻址系统，到目前为止仍然能够正常工作真可以说是一个奇迹。

因为 Internet 地址资源正在迅速被耗尽，大多数的家庭网络都使用网络地址转换（NAT）

技术建立了一个网关。NAT 是 Internet 工程任务组（IETF）制定的一种标准，它允许私有网络中的多台 PC 或设备共享一个全球唯一的公共地址(所使用私有地址的范围为 10.0.x.x、192.168.x.x 和 172.x.x.x）。作为对 IP 地址短缺的一种临时补救措施，NAT 可以很好地完成很多工作。例如 Windows XP 的 Internet 连接共享就使用 NAT，就像很多网关设备（例如 DSL 和线缆调制解调器）所做的一样。

问题是，NAT 希望所有的网络应用程序都以一种标准方式（即在数据包头中使用 IP 地址）进行通信，但是有些网络程序预计到 NAT 的存在，它们使用了 NAT 无法转换的嵌入式 IP 地址。

2．NAT穿越技术

NAT 穿越技术允许网络应用程序，对它们是否位于一个具有 UPnP 能力的 NAT 设备之后进行检测。然后，这些程序将获得共享的全球可路由 IP 地址，并且配置端口映射以将来自 NAT 外部端口的数据包转发到应用程序使用的内部端口上。所有这一切都是自动完成的，用户无须手动映射端口或者进行其他工作。NAT 穿越技术允许网络设备或者点对点应用程序通过动态开启和闭合，与外部服务之间的通信端口穿过 NAT 网关与外界通信。

5.7.4 使用 UPnP 的基本条件

使用 UPnP 并不是任何情况下都可以的，需要满足一定的条件才行，基本来说需要满足下面 3 个条件。

1．硬件支持

很多用户都是通过 Modem 来连接外网的，所以 Modem 在硬件要求上要支持 UPnP 功能才能使用 UPnP，确定此 Modem 是否具备此功能可查阅说明书或者直接咨询厂家。一般来讲，Modem 还必须同时支持路由功能，除非你配备了单独的路由器。

2．软件支持

要使用 UPnP 功能的，那么其对应软件也必须支持 UPnP 功能，这是很简单的道理。当前的很多 P2P 软件都支持 UPnP，常用的如 BitComet、BT、eMule 等软件都支持 UPnP 功能。

3．操作系统支持

操作系统指的是运行 UPnP 的系统平台，这是最基本的要求。当前的系统平台有很多，但微软的官方网站声称从 Windows Me 开始就已经支持 UPnP 功能了。至于 Linux、UNIX 下是否支持，读者可以自行测试。

5.7.5 设置 UPnP 的支持

满足了以上 3 个基本条件，而且也有使用 UPnP 的必要，那么就可以使用 UPnP 了。设置 UPnP 的支持也很简单，基本上完成以下几个操作就可以了。

1．在Modem中打开UPnP功能

不同型号的 Modem 设置界面和方法略有不同，如有些是在下拉菜单中选择 Enable，但基本的原理都是一样的。

进入路由器的设置界面，如果你的路由器支持 UPnP，那么在转发规则选项卡下就会看到 UPnP 设置选项（不同路由器可能会有不同），如图 5.18 所示。在此选项中，只需选择启用 UPnP 即可。然后重启一下路由器就完成了路由器的设置。



图 5.18　在路由器中开启 UPnP 功能的设置界面

2．在操作系统中打开UPnP功能

在 Modem 上设置好了 UPnP 的技术，就可以在系统中使用 UPnP 的相应功能了。如果你使用的是 XP SP2 系统，可按如下步骤进行设置。

（1）单击“开始”|“设置”|“控制面板”，在控制面板的列表中选择“添加或删除程序”选项。在弹出的对话框中单击“添加/删除 Windows 组件”，会弹出 Windows 组件向导窗口，如图 5.19 所示。

（2）在图 5.19 所示的界面中，选中“网络服务”选项，然后单击“详细信息”按钮，弹出“网络服务”对话框，如图 5.20 所示。



图 5.19　Windows 组件向导界面

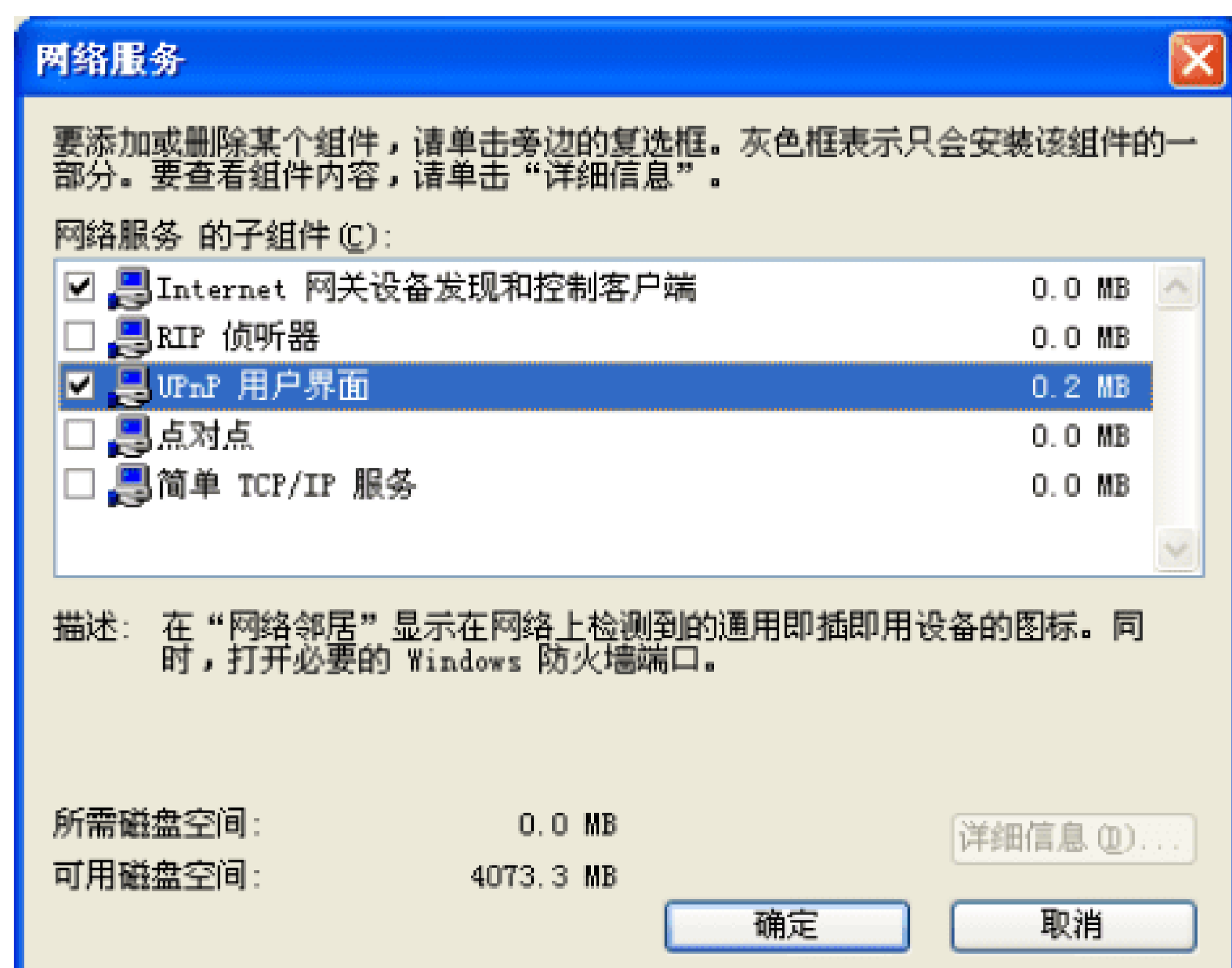


图 5.20　配置 UPnP 用户界面

（3）在图 5.20 所示的界面中，选中“UPnP 用户界面”即可，单击“确定”按钮。

然后，系统会自动安装相应的组件，可能会提示插入安装光盘，总之按照提示操作完成即可。

完成了对系统的设置，还需要配置 Windows 自带的防火墙。设置方法也很简单，只需打开 Windows 自带的防火墙，在“例外”选项卡中选择“UPnP 框架”即可。具体的操作方法如下。

（1）单击“开始”|“设置”|“控制面板”，在控制面板的选项列表中，双击打开“网络连接”选项。

（2）在弹出的“网络连接”对话框中，右击选择本机上网所用的网络连接，在弹出的快捷菜单中选择“属性”选项。

（3）接着会弹出一个“本地连接属性”对话框，选择其中的“高级”标签并单击窗口下方的“设置”按钮。

（4）在弹出的“Windows 防火墙”对话框中，选择其中的“例外”标签，然后在其中选择“UPnP 框架”即可，如图 5.21 所示。

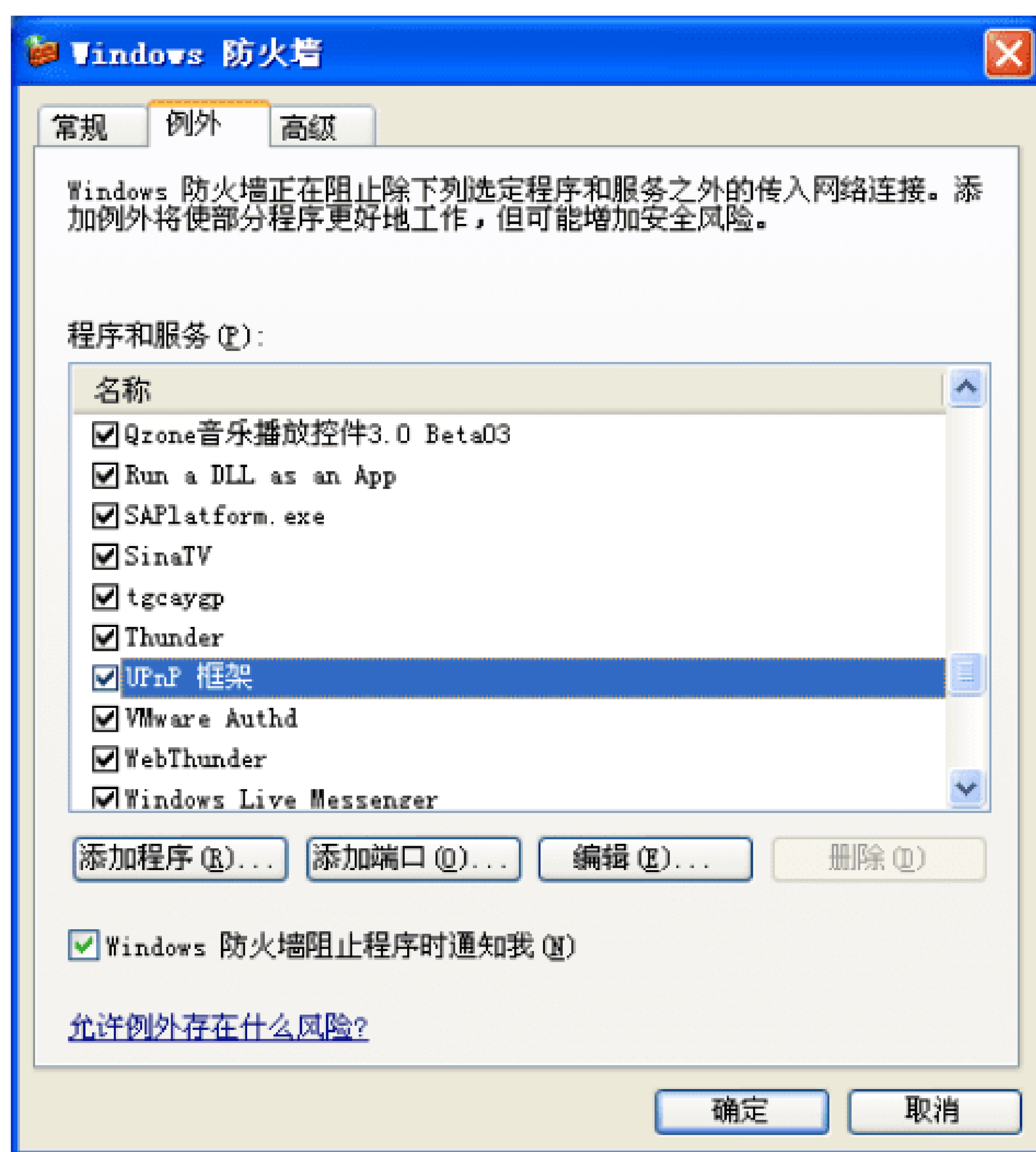


图 5.21　启用 UPnP 框架界面图

其实有个更加简单的方法可以同时完成以上两步。双击桌面上的网上邻居（注意是鼠标左键双击，不是右键查看属性），然后单击左侧菜单列表中“显示联网的 UPnP 设备的图标”，系统会自动安装 UPnP 组件以及在防火墙中打开 UPnP 框架，实际上就是一次性完成上面两步的工作。

如果使用的是 XP SP1 系统，那么在“Windows 组件”中显示的是“通用即插即用”，而不是“UPnP 用户界面”，选择此项即可。而且 XP SP1 系统的防火墙并没有 UPnP 框架的选项，需要手动进行端口添加，设置方法简要描述如下。

在防火墙设置中，单击“高级”选项卡，然后自行添加两个端口，TCP 端口类型，端口号为 2869；UDP 端口类型，端口号为 1900。由于使用了 NAT 网关，所以应该设置的是

连接到该网关的网卡防火墙。而且网关内部均为内网，所以开启这两个端口，不会对系统造成安全隐患（除非 NAT 网关被绕过，否则外部连接无法检测到该端口）。

以上的防火墙设置只是针对 Windows 自带的防火墙，如果安装了其他的防火墙，必须在该防火墙中打开 UPnP 框架。

3．在Windows中打开相应的UPnP服务

完成了以上的设置，就可以在系统中打开 UPnP 的相应服务了。进入控制面板，选择“管理工具|服务”，从服务器列表中，找到 SSDP Discovery Service 和 Universal Plug and Play Device Host 两个服务选项。右键单击相应的服务项，从弹出的右键列表中选择“属性”，在弹出的“属性”对话框中，分别启动这两项服务。

做完以上工作后，如果操作正确，我们就可以在“网络连接”中看到多了一项网关，这表明添加 UPnP 已经成功。

4．打开P2P软件中的UPnP功能

以 eMule 为例来说明如何打开 P2P 软件中的 UPnP 功能。启动 eMule 后，在 eMule 的工具栏中单击“选项”选项，在“选项”对话框的左侧菜单中选择“扩展设置”选项。然后分别选择“使用 UPnP”、“在 UPnP 中，尝试使用随机端口”这两项设置，设置界面如图 5.22 所示。

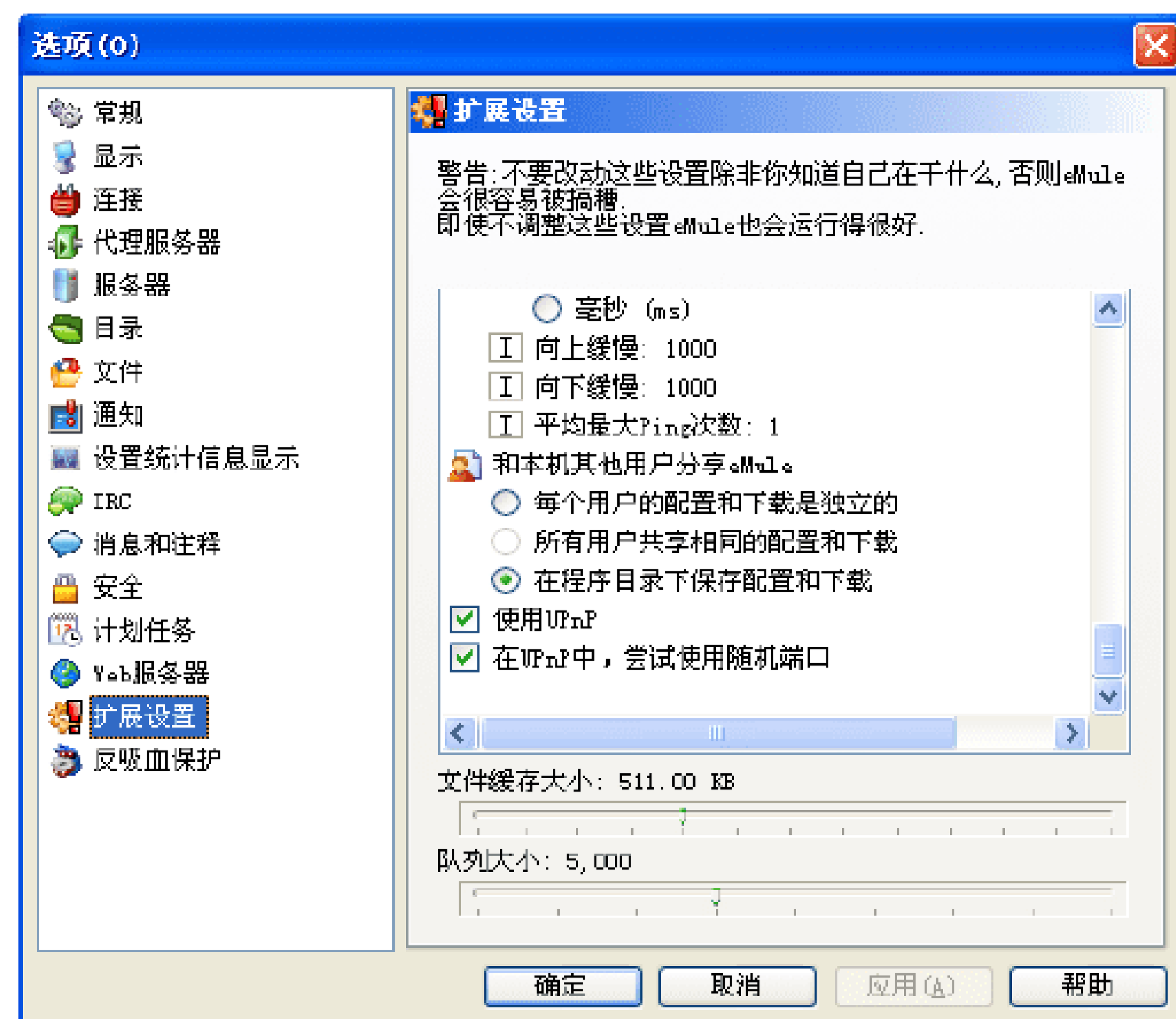


图 5.22　在 eMule 中开启 UPnP 功能的设置界面图

以上设置完成后，即使在内网，再使用 eMule 下载时，连接服务器后就会显示为高 ID，如图 5.23 所示。显示本机的 IP 地址是内网 IP，但 eMule 的网络连接显示的是 HIGH ID 的特征。证明 UPnP 起作用了，关于以上 UPnP 的设置也是成功的。

由图 5.23 可知，从框线标注的部分可以清楚地看到，IP 地址是 192.168.1.XXX 形式的内网址，在 eMule 的网络信息中，显示的是 High ID。到此为止，打开 UPnP 支持并在 P2P 应用系统中使用 UPnP 才算完成。

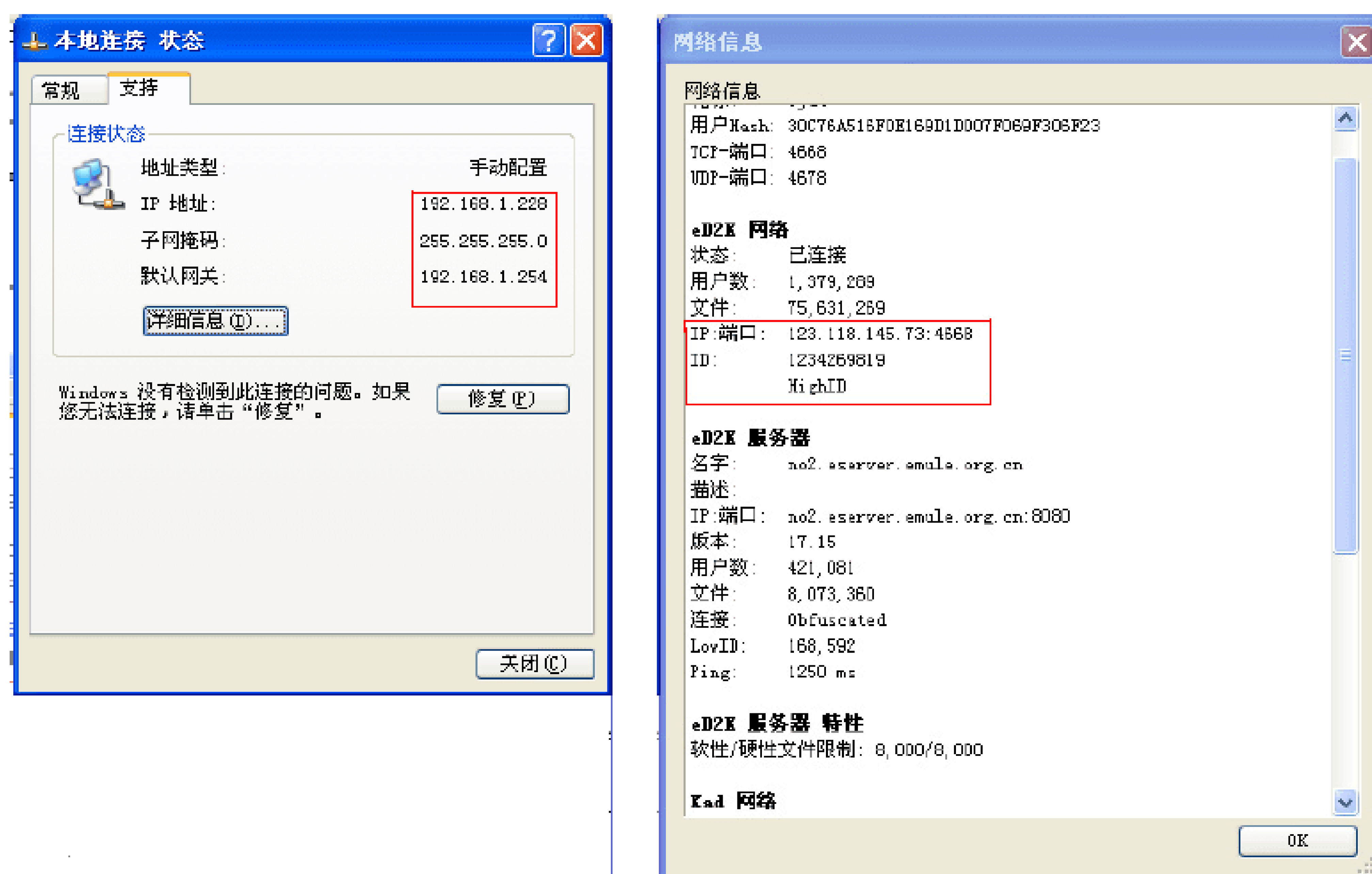


图 5.23　在内网中查看 eMule 的 High ID 界面图

注意：在 eMule 中查看高 ID 还是低 ID 的方法是在工具栏上任意地方右击，在弹出的快捷菜单中选择“自定义”|“工具条”就可以调出服务器按钮了，然后单击服务器按钮就可以看到。如果是 easymule 版本的话，在分享按钮旁，单击一个向下的箭头点，然后选择高级选项，也可以打开服务器界面。另外，在软件的状态栏上有个小小的地球图标，有一上一下两个箭头，也是显示 ID 状况的，其中绿色表示高 ID，黄色就是低 ID，而红色表示没有连接到网络。

其他的 P2P 系统如 BT、BitComet 等也都有类似的设置，读者可自行尝试。

5．UPnP映射失败的原因

在设置 UPnP 功能支持的过程中，也经常会发现 UPnP 映射失败的情况。UPnP 设置失败潜在有很多种原因，常见的有以下 3 个原因，如果发行设置失败的情况，可对照进行排查。

- 系统服务中禁止了 SSDP 服务（用于寻找 UPnP 设备），按照以上所说的，开启 SSDP 服务功能即可。
- 开启了 XP 下的 SP1 的 ICF，就是开启网络连接防火墙。只需关闭防火墙及相应的设置即可。

注意：XP 的 ICF 与 UPnP 设备发现有冲突，SP2 修复了这个问题，但是仍然需要在防火墙设置中允许例外，就是 UPnP 框架。

- 路由器不支持 UPnP，请向制造商询问，或者更换支持 UPnP 的路由器。

以上就是对 UPnP 的简要说明，在这里总结一下就是，UPnP 通过端口映射的功能，允许基于 P2P 技术的应用系统打通局域网限制，在安全的前提下，连接上更多的 Peer 来加速 P2P 网络中 Peer 之间的连通性并扩大交互的范围。UPnP 在 P2P 网络互联、P2P 通信及 NAT 的穿透方面有重要的应用意义。

5.8　本 章 小 结

本章重点讲述了 P2P 网络中 NAT 穿透的相关技术，从 NAT 的基本知识体系到 NAT 的工作原理、NAT 的穿透方法等多方面都作了详细的阐述，还以实例的形式讲述了 NAT 穿透在 P2P 系统中的应用。NAT 穿透是 P2P 网络的重要知识点，对 P2P 网络中 Peer 的连通性、网络的交互性、数据传输、资源的共享规模都有重要的影响，尤其是在当前基于 IPv4 的网络体系下，研究 NAT 的穿透技术更有重要的现实意义。

通过本章的学习，读者进一步了解了 NAT 的整个技术体系结构，理解 P2P 网络中 NAT 穿透的原理和基本方法，并能根据这些原理知识实现 P2P 网络中简单的 NAT 穿透模型。

第 6 章　基于 P2P 的 BT 技术解析

BT 模型以无结构 P2P 模型为基础，结合相关传统网络技术（例如 HTTP 协议），为文件高效安全地分发提供一个稳定的应用平台。BT 模型广泛应用于文件分发和文件下载。在国内所有的 P2P 应用体系中，基于 BT 模型的应用占据了大半的用户群。同时，由于 BT 模型提供文件高速分发，也逐渐开始应用于网络流媒体系统，例如视频点播、媒体广播等。

BT 特性及其在互联网上的广泛应用，使得 BT 在 P2P 技术体系中占用重要的地位。本章就针对 BT 技术进行详细的讲解，以 BT 协议规范为核心，重点讲解 BT 的工作原理、协议分析、模型系统以及重要的算法实现，使读者对 BT 技术有一个全面的理解，能够在本章讲解的基础上进行 P2P 基本应用的模型构建和系统开发。本章讲解的知识点如下。

- BT 概述：主要讲述 BT 的基础知识及 BT 下载与传统下载方式的比较，初步了解 BT 下载异于常人的地方。
- BT 下载技术：系统地讲解 BT 是如何实现下载的，重点掌握 BT 工作原理及下载过程的实现，还要了解它是如何部署下载的。
- BT 协议规范及分析：详细分析 BT 协议规范的核心内容，重点掌握 BT 编码的规则、元信息文件的结构、BT 与 Tracker 通信、BT 用户之间的通信等。
- 如何使用 BT：了解常用的 BT 客户端软件，会使用 BitComet 进行 BT 下载，会对 BT 下载的过程、状态及相关参数进行配置。
- BT 的影响及发展：了解 BT 的双面效应，了解当前 BT 的研究热点和发展的趋势。

BT 是 P2P 技术的经典应用，学习本章，是学习 P2P 技术实战开发的基础，要充分理解 P2P 是如何在 BT 技术中得到完全体现的。学完本章，要开发一个简单的 BT 下载客户端程序将不再神秘。

6.1　BT 概述

BT 是一个多点下载的源码公开的 P2P 软件，比特流是一种内容分发协议。它采用高效的软件分发系统和点对点技术共享大容量文件，并使每个用户像网络重新分配结点那样提供上传服务。基于 P2P 技术的 BT 系统，可以使下载服务器同时处理多个文件的下载请求，而无须占用大量带宽，在实际应用中常被人们称之为“群集、散布、集中”的文件传输协议。最初，它由程序员 Bram Cohen 使用 Python 语言编写，并且还是代码开源的专利软件，可以自由地下载和传播。目前，各种支持 BT 下载的软件层出不穷，BT 技术及思想在商业和科研中也有广泛的应用。本节将就概述性地讲解 BT 的基础知识。

6.1.1　BT 简介

电影公司恨它，文件交换者爱它，布拉姆·科恩的超快 P2P 软件，已经把国际互联网络变成了一个世界范围内自选预定交互式的电视系统。想要免费的录像点播吗？想要无奇不有的网络资源吗？在 BT 上点击一下就行了。那么 BT 是什么呢？为什么它有这么大的能耐？

BT 俗称 BT 下载、变态下载，是一个多点下载的源码公开的点对点软件，应用广泛，使用方便。

Bram Cohen 是 BT 下载的创始人，如图 6.1 所示，就是年轻时的 Bram Cohen。由 Cohen 创造的 BT，是迄今为止最成功的 P2P 系统之一，它可以让用户快速地上传和下载有庞大数据量的各类文件。普通下载方式需要几个小时才能下载完的文件，但用 BT 下载却能在数分钟内搞定。据不完全统计，目前已经有近六千万人下载过 BT 的应用程序，BT 在整个互联网上的流量占了三分之一强。这些数据足以说明 BT 在网络应用中的重要地位。



图 6.1　BT 之父——Bram Cohen

注意：了解一下 BT 之父——Bram Cohen。

Bram Cohen 生于 1975 年，他曾在几家网络公司担任过计算机程序员，而真正使他声名远扬的是他编写的 BitTorrent 软件。其实 Cohen 患有 Asperger syndrome（阿斯伯特综合症，即轻孤独癖），这使他具有高度的集中力，但在社交上却有障碍。

BitTorrent 对于 Cohen 来说，一直是一种脑力训练而不是一种赚钱的途径。不像其他文件交换程序，BitTorrent 不但是免费的，而且还是开源的。所以尽管 BitTorrent 获得了巨大的成功，然而，直到 2005 年下半年，它都没有为 Cohen 带来过一分钱。其实一直以来，Cohen 通过自己的网站 bitconjurer.org 能接收到 BitTorrent 用户的捐款，但是这笔金额一直很少。

目前，BT 涉及到大规模侵犯版权问题，促使美国电影协会开始向 BitTorrent 站点的管理人发出了侵权通知。对此 Cohen 表示自己开发这个系统的初衷只是为了使人们在购买合法在线音乐时，不需要再经历那么漫长的等待。他表示，自己对所创造的这个系统早已失去了控制。

BT 是一个开放源码系统，Cohen 于 2002 年在一次电脑黑客会议上展示了他的编码，这些编码作为一种免费、开放式的源码程序，以 Python 写成，以 MIT 许可证发布。

在设计之初，BT 主要用于对大型文档和自由软件如 Linux、FreeBSD 的发布。对于发布数百 MB 以至数 GB 的档案时，如 Fedora 的光盘镜像格式，BT 的使用能大大减低服务器的数据流量从而减低发布的成本。另外，一般有新版本软件推出时，服务器必定人山人海，使用 BT 也能大大减低繁忙时间服务器的负担。而现在，BT 几乎应用在任何文件的发布方面，不论文件大小、文件类型、文件格式等，都可以用 BT 进行分发。

6.1.2　BT 下载描述

BT 下载可以用 3 个简要的特征来描述：

- BT 下载是多源发送的，下载源有一个或多个，也可能没有。
- BT 在下载的同时，也进行上传。
- 下载的人越多，速度越快。

BT 下载，不像常规的 Web 下载那样只有一个发送源，BT 可以有多个发送点，当你在下载时，同时也在上传，使所有的 Peer 都处在交织通信、同步传送的状态。BT 下载的过程，可以用以下几步来简要说明。

（1）BT 首先在上传一端把一个文件分成了多个部分，客户端甲在服务器随机下载了第 N 部分。

（2）客户端乙在服务器随机下载了第 M 部分。

（3）这样，甲的 BT 就会根据情况到乙的电脑上去拿乙已经下载好的第 M 部分。

（4）乙的 BT 就会根据情况去到甲的电脑上去拿甲已经下载好的第 N 部分。

通过这种方式就可以大大加速下载速度。从原理上来说，由于 BT 先进的下载方式减轻了服务器端的负荷，同时加快了客户端的下载速度，所以，用 BT 下载的人越多，速度越快。

6.1.3　传统下载方法与 BT 下载的比较

上文对 BT 的下载方式进行了简要的说明，可以看出，BT 下载与传统的下载方法有明显的不同。为进一步理解 BT 下载特点，下面就对传统下载方法与 BT 下载方法进行比较。

下载的定义就是把服务器一端的数据传送到客户机一端，一般来讲，像 FTP、HTTP、PUB 这样的下载方式都是用传统的方式进行下载，它们的下载原理就是将数据放到一个中央服务器上，需要下载的客户端都连接到这个服务器，然后从这个服务器上读取数据，工作原理如图 6.2 所示。

这种下载方式通过服务器将数据分发到各个客户端，虽然也能达到下载的目的，但是这样就出现了一个问题。随着用户的增多，对带宽的要求也随之增多，对服务器的性能要求也会增高，一旦超过一定的限值，就会下载瓶颈，造成服务拥堵、下载速度剧减，甚至会造成服务器的生死机。所以，运用传统的下载方式，很多的服务器都会有用户人数的限制，下载速度的限制，这样显然会给用户造成很多的不便。

但 BT 不同，用 BT 下载的特点之一就是用户越多，下载速度越快，这是为什么呢？

因为 BT 用的是一种类似“传销”的方式来达到共享，工作原理如图 6.3 所示。

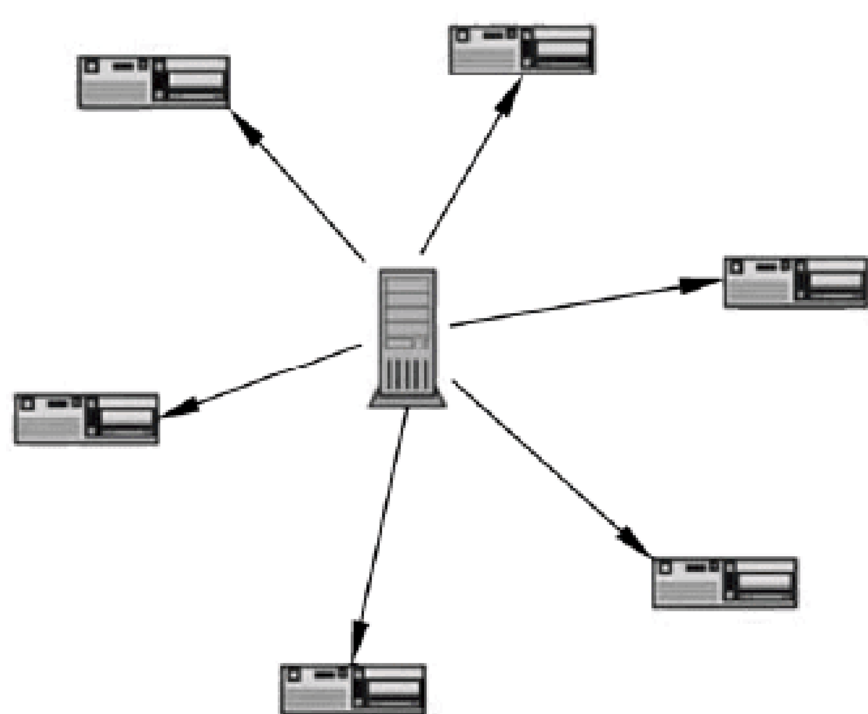


图 6.2　传统的 HTTP、FTP 的下载原理

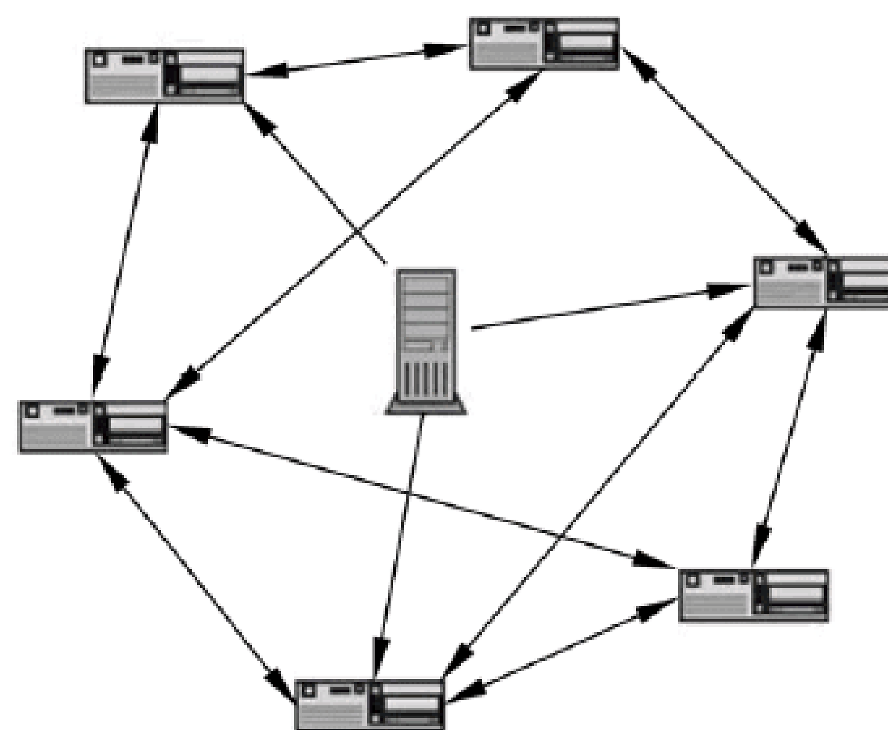


图 6.3　BT 方式的下载原理图

从图 6.3 中可以看出，作为服务器的 Peer 把文件传给其他的 Peer 后，各个 Peer 之间也进行交互传。这样，单一的下载源就变成了多个源，Peer 越多，彼此之间交互的范围和数量也就越大，下载速度也就越快了。这个过程可以下面的实例来说明。

当用 BT 下载一个几百 MB 的文件时，首先，把这个几百 MB 的文件进行分解，分成一个个有特殊标记的分片，然后让不同的 Peer 下载不同的分片到各自的电脑上。这样，一个 Peer 所拥有的单个文件，就被分散到其他各个 Peer 上了，但所有这些分散的 Peer 上所存储的都不是完成的文件。然后在这些分散的 Peer 之间，互相分享各自拥有的部分，直到每个 Peer 都拥有一个完整的文件为止，这样整个下载过程也就完成了。

用 BT 下载可以用以下 3 步来描述。

（1）一个文件分成了 3 个部分，甲下载了第一部分，乙下载了第二部分，丙下载了第三部分。

（2）甲下载完第一部分后，就可以脱离与服务器的交互，直接与乙和丙连接，从乙那里下载第二部分、从丙那里下载第三部分。当甲同时下载完这三部分后，就可以将这三部分进行组合，形成一个完整的文件。

（3）依次类推，乙可以从甲那里获得第一部分内容、从丙那里获得第三部分内容，而丙从甲取得第一部分，从乙取得第二部分。

这样，每个 Peer 都可以通过 Peer 之间的交互而取得全部文件的内容，Peer 之间一直这样交互下去。每个 Peer 都与那些尚未下载到有关部分的其他 Peer 分享它们已经下载的部分，直到全部下载完成。

BT 这种下载方式，简而言之，每个人既是客户，又是分销商，大家与其他人共享文件，并分担在互联网上传输文件的工作，这就是 BT 令人称奇的高效工作，它分解了传输文件的任务。这意味着并非每个人都要一口气从一个地址下载全部的文件，从而缓解了试图传输这份文件的人的压力。而且，这也意味着每个人获得文件的速度要快得多，因为他们将下载的工作分散在十几台电脑上。

BT 的原理是下载的人越多，速度越快，完全不同于以往的任何同类软件。而传统 Http、FTP 的服务器下载方式，速度要取决于你的带宽和服务器分给你的带宽。

注意： 下载的人越多，速度就越快，这只是一种表面的现象，并不是总体的网络负载减轻了，而是通过 BT 技术，将庞大的网络负载均衡到每个 Peer 中。当下载的人多时，每个 Peer 均衡的负载就会变小，下载来源会变广，因而，直观的感觉就是速度也就越快了。读者只要理解了 BT 的原理，这句话的真实意义也就不难理解了。

6.2　BT 的下载技术

BT 下载技术是一个广义的概念，它并不是简单地指一个下载的过程，而是从 BT 下载协议、BT 下载部署到 BT 下载实现这一整套流程，本节就对 BT 下载技术的主要方面知识进行讲解。

6.2.1　BT 下载的部署

根据 BT 协议规范，要完成一个 BT 下载过程，至少需要一个静态的“元信息”文件，一个跟踪（tracker）服务器和终端下载者。这里，终端下载者指的就是用户的 PC，也就是 BT 的下载客户端，终端下载者已经确定的情况下，BT 完成一次下载部署，还需要一个 tracker 和一个“元信息”文件。

所谓 tracker，指的就是一个服务器，负责帮助终端用户之间相互建立连接，终端用户就是 P2P 网络中所指的 peer。“元信息”文件，就是通常所说的.torrent 文件，也叫 seed，被称为“种子”，是被下载文件的拥有者。

BT 是通过一个扩展名为.torrent 的文件进行下载部署的，.torrent 的文件放在一个普通的 Web 服务器上，是一个普通的文本文件，大小只有几十 KB，类型固定，数据结构严格，用户可以在相关 BT 发布网站上自由下载。当要下载 BT 网络中的资源时，必须要先找到描述这一资源的.torrent 文件，此文件包含了要共享的文件的信息，包括文件名、大小、文件的散列信息和一个指向 tracker 的 url。

得到.torrent 文件后，终端的下载者通过使用 BT 客户端打开.torrent 文件，读取.torrent 文件内容，取得与 tracker 服务器进行通信的相关信息，就可以开始下载了。BT 完成一次下载部署的完整过程可描述如下：

（1）位于终端的第一个 seed，将自己拥有的文件资源，按照 BT 协议规范中 B 编码的要求，将资源文件的各类信息写入到一个.torrent 文件中，然后将此文件发布到普通的 Web 网络中，供需要此资源的人下载。

（2）第一个 seed，打开 BT 下载终端，向 tracker 服务器注册，等待为别人提供文件。这样一个 seed（种子）就注册到 tracker 服务器上了。

（3）第一个 Peer 要下载 seed 发布的资源时，在互联网上搜索到描述此资源的.torrent 文件。正确地打开.torrent 文件后，根据.torrent 文件中描述的 tracker 信息，向 tracker 服务器注册，这样，第一个 peer 完成向 tracker 的注册，并取得 seed 的信息。

（4）peer 与 seed 建立连接，告诉 tracker 自己要下载的文件、自己使用的端口以及类似的信息，并从 seed 处读取文件。由于原始的文件只有 seed 拥有，所以 seed 至少要上传原始文件的一份完整备份。

（5）当有另外一个 peer 加入进来后，tracker 负责帮助新加入的 peer 获取 seed 和其他 peers 的信息。新 peer 利用这些信息和 seed 及前一个 peer 建立连接，然后从这两者处获取文件，并上传自己拥有的文件片段。

（6）反复执行步骤（3）和步骤（4）这样的过程，直到所有的 Peer 都完成文件的下载

为止。

以上的交互过程，可以用图 6.4 来表示。

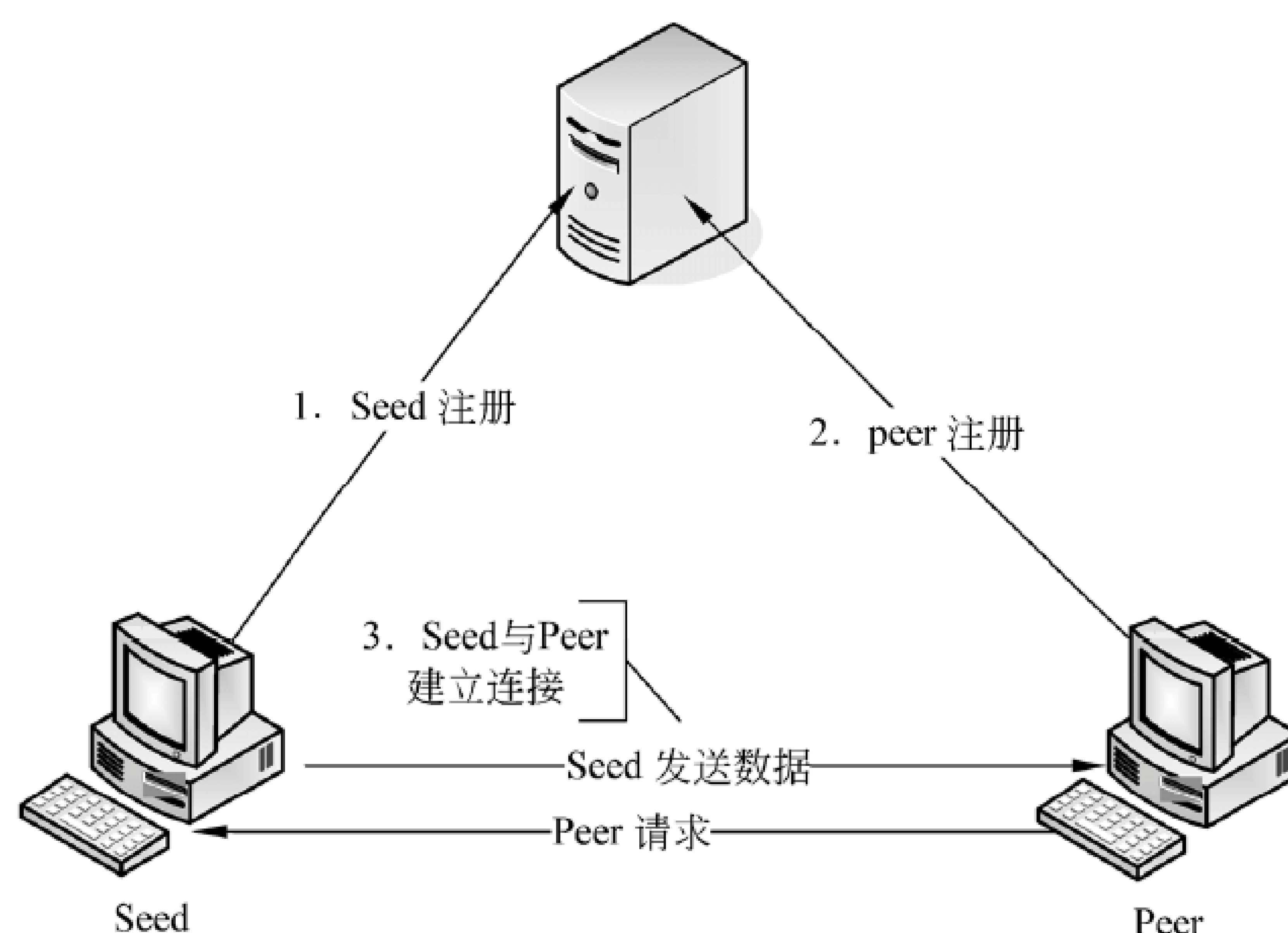


图 6.4　BT 完成一次下载的部署过程

注意：BT 下载的实际过程并不是以上所说的那么简单，在实际下载中还要处理 Sha1 校验、断点－续传、文件分片、结点控制等，在下文这些都会讲到。

6.2.2　BT 工作原理

BT 中文件下载相关的逻辑问题，通过 peers 之间的交互传递资源来解决。BT 工作的原理如图 6.5 所示。

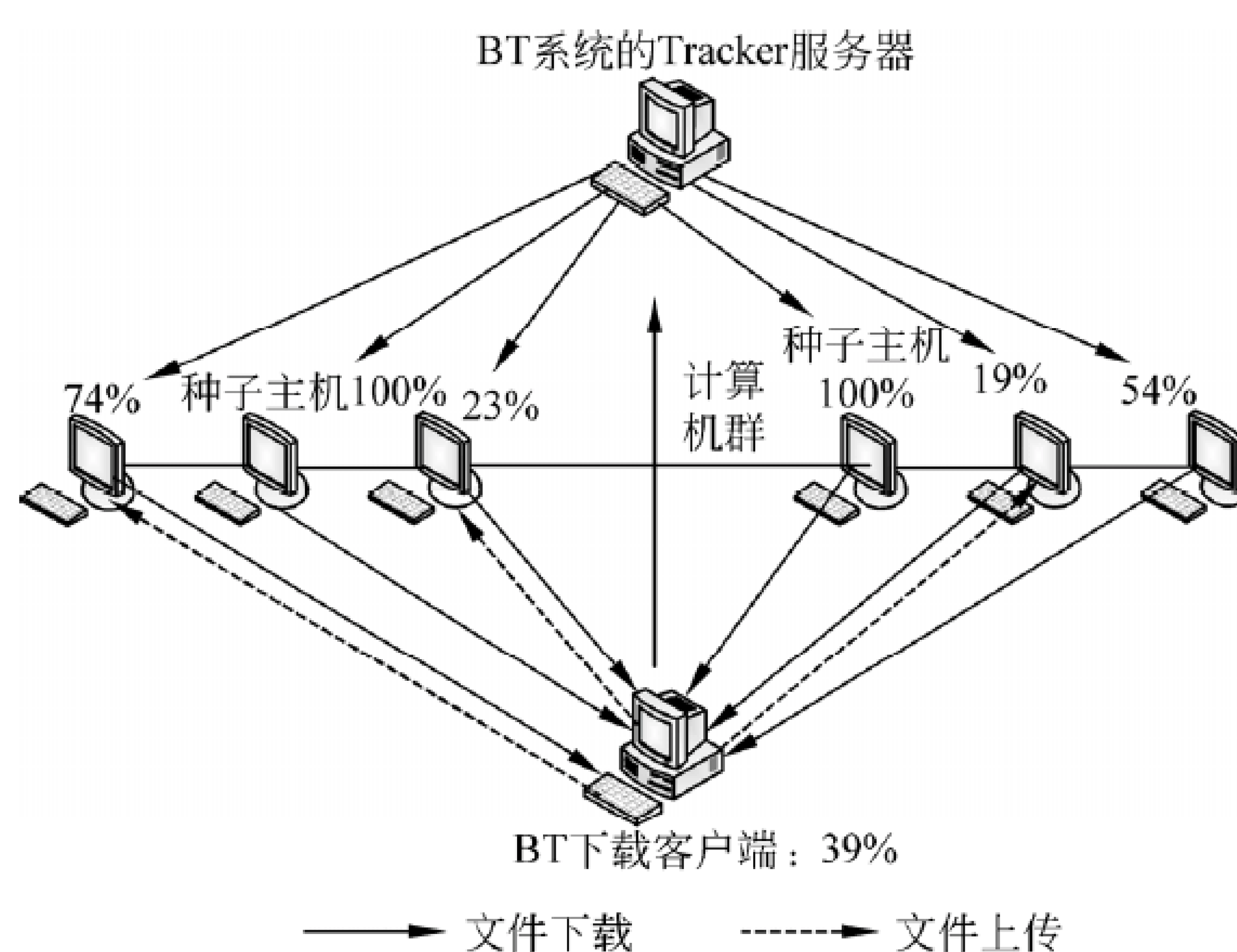


图 6.5　BT 下载的工作原理

从图 6.5 中可以看出，在 BT 下载中，各个 Peer 之间是对等的，下载过程是渐近流水

式的，下面就重点讲一下 BT 的下载原理。

1．Peer对等原理

在BT下载部署过程中已经清楚的知道，整个参与下载的角色只有下载终端也就是Peer和 Tracker。而 Tracker，只保存一些关于 Peers 下载和上传速率的信息，它的职责被严格限定为“帮助 peers 相互发现对方”。因而，Tracker 不实际参与文件的传输，本身也不保存要下载文件的任何信息。

那么，所有参与文件下载和上传的过程都在 Peer 间进行。在 BT 网络中，没有专门的供文件下载的服务器，参与文件交互的 Peer，即充当服务器的角色，也充当客户端的角色，在进行文件下载的同时，也负责上传文件，Peer 与 Peer 之间是双向对待的关系。

2．BT的工作原理

BT 的工作是从解析元信息文件（.torrent 文件）开始的，从.torrent 文件里得到 Tracker信息，然后与 Tracker 交互得到 Peer 信息，在 Peer 间进行交互实现下载，整个工作原理可用以下几个要点来描述。

（1）.torrent　的作用

使用过 BT 系统的人都知道，要用 BT 下载，就要先下载一个.torrent 文件，这个文件到底有什么呢？首先，announce 记录了发布服务器的位置，让 BT 知道是哪个 Web 服务器发布的，然后是一些文件信息、文件名、目录名、长度等，最后是片段长度和片段的 Sha1校验码。以文本文件的形式打开.torrent 文件，就可以知道文件的大概内容，文件中有些内容是乱码，那些都是文件片段的 SHA-1 校验码。

注意： BT 为了实现续传和文件校验，就把文件分成若干个片段，需要用 SHA-1 校验法来检验文件的完整性和正确性。

（2）开始-续传的实现和 SHA-1 校验

BT 打开一个.torrent 文件后，要先选择文件保存在哪里。然后判断文件不存在的话就建立新文件，存在的话就用 SHA-1 校验码去校验文件。如果校验错误，说明是还没下载的文件，这样就可以实现续传了。

（3）得到 peer

现在知道要下载什么了，但要到哪里下载呢？这就要寻找谁可以提供上传了。这里 BT是通过 Web 服务器来实现的，首先 BT 会通过分析 torrent 来得到下面一串网址：

http：//btfans.3322.org：6969/announce?info_hash=%CDg%D4%19%AD%96%9D%93%03%DB%E4%FFXA%C6%5D%043%17O&peer_id=%00%00%00%00%00%00%00%00%00%00%00%00%A3E%E0%9BeB%90d&port=6882&uploaded=0&downloadd=0&left=19171922&event=started。

下面来分析这一串代码。

- http：//BTfans.3322.org：6969/announce　是发布服务器的地址。
- info_hash：是 torrent 文件中 info 部分的 Sha 校验码，Web 通过它在发布列表找到对应的记录。
- peer_id：自身的标识，它是 12 个 0 和当前时间+全球唯一标识码（GUID）的 Sha

校验的前 8 位，共 20 位。

- port：提供上传的端口号。
- IP：你的 IP 地址，没有的话服务器会自己找到。
- uploaded、downloaded：你上传和下载了多少，服务器可以用它来做流量分析。
- left：还要下载多少个字节。
- event：状态，告诉服务器你是准备开始下载，还是停止，或下载完成了。

以上就是根据 torrent 文件内容分析出来的一些信息，这种操作默认 5 分钟做一次，或由服务器设定。

注意：这段代码为什么这样分析，代码中间出现的术语为什么要这样表达？这些知识现在先做简单了解，详细的说明，请参考本章后面所讲的 BT 协议分析部分知识。

（4）服务器会做什么

服务器中有一个 track 程序来管理这些请求，得到这一串代码后就会用 info_hash 来查找列表，找到了就可以下载，找不到就无法进行对等连接。与此同时，Tracker 服务器还会根据 IP 和 Port 进行反向连接，以测试此用户是内网用户还是公网用户。然后服务器返回现在正在下载这个文件的所有公网用户的 IP（IP 地址）和 Port（端口），同时，会将公网用户的 IP 和 Port 放到 info_hash 对应的列表中，并更新列表。这样当有其他的 BT 客户端加入时，就可以通过 Tracker 服务器查询到可提供下载源的公网 IP 和 Port。

Tracker 服务器返回的信息也是以 B 编码的形式表示的，根据 B 编码规则解析 Tracker 服务器的返回信息，就能确定当前正在下载的活跃 Peer 结点。

注意：Tracker 服务器在反向连接的时候，如果是内网用户，是无法连通的。因为内网用户的 IP 和端口，是内网出口的 IP 地址和端口，显然无法和内网主机中的 BT 客户端进行连接。

在 Tracker 服务器的返回信息中，其中有一个时间参数，用来告知 BT 客户端用户隔多少时间来查询一次 Tracker 服务器。因为 BT 网络的动态性非常高，在 1 秒钟内，也有可能会有成百上千个 Peer 结点加入或退出，所以需要一个时间参数来设置查询 Tracker 的频率。

（5）下载

得到这些 peer 的 IP 和 Port 以后，BT 客户端就可以根据对应的 IP 和 Port 连接到对等的、有下载源的 Peer 结点上，这样两个对等的 Peer 结点之间就可以进行资源的交互和传输。

BT 的某一客户端结点在与其他的 Peer 连接过程中，会到所有的存在下载源的 Peer 结点上去寻找自己要下载的资源，每找到一个 Peer 结点就与之建立一个 Socket 连接来进行下载。所以下载的人越多，Peer 的活跃数就越多，下载速度也就越快。

经过以上（1）～（5）步的操作，从理论上讲，就可以实现 BT 的下载了。

6.2.3　BT 的下载实现

在上文中，详细地分析了 BT 的下载流程。下面就根据这一流程，讲解 BT 下载到底

是如何实现的，怎样通过 Peer 与 Peer 之间的连接把一个分片的文件完整地下载到本地。

1．BT下载的执行过程

下面就用几幅图片来形象地描述 BT 下载的完整执行过程。以下的过程描述都是假定，在 torrent 文件已知、种子资源文件存在、tracket 服务器、本地客户机都已部署好的理想状态下进行。

（1）作为种子的 Server，存储有一个完整的文件，这个文件分为 4 个块（4 pieces），这 4 个块就是文件的分片，分别为红片、蓝片、绿片和黄片。有 4 个 Client 通过 BT 的方式下载这个文件。刚开始的时候，每个 Client 都得到这个文件其中之一的分片，如图 6.6 所示。

（2）得到分片后的 4 个客户机，不再与服务器交互，直接通过 Client←→Client 之间的交互相互之间下载资源，如图 6.7、图 6.8、图 6.9 和图 6.10 所示。

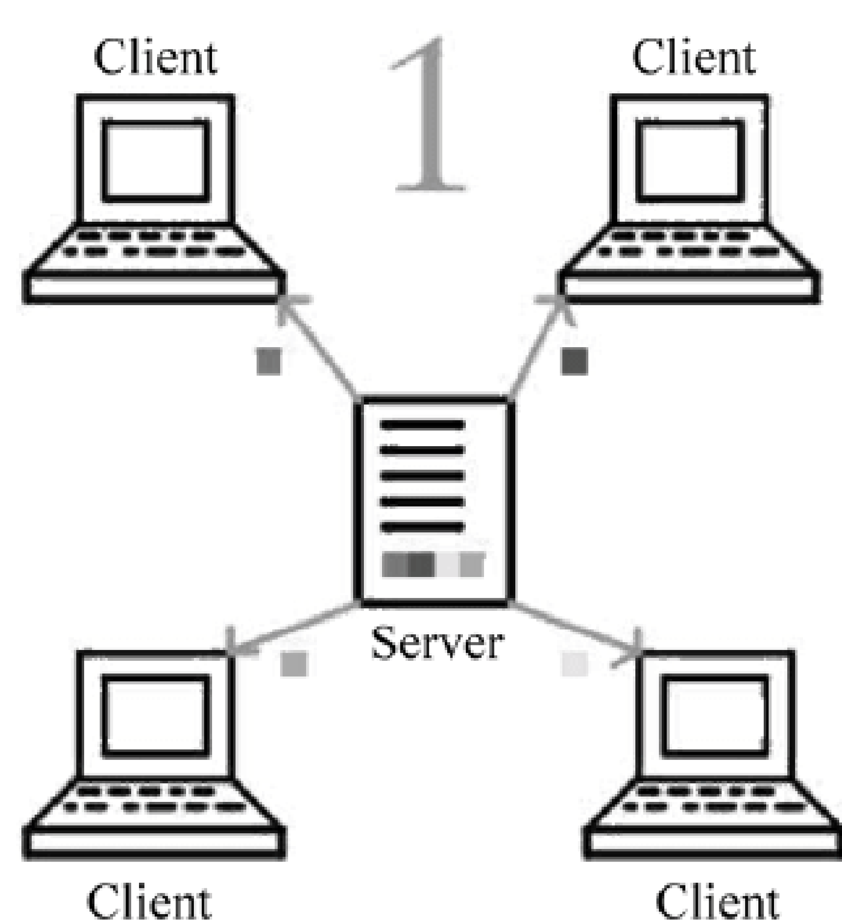


图 6.6　作为种子主机将文件的 4 个分片分别传给 4 个客户机

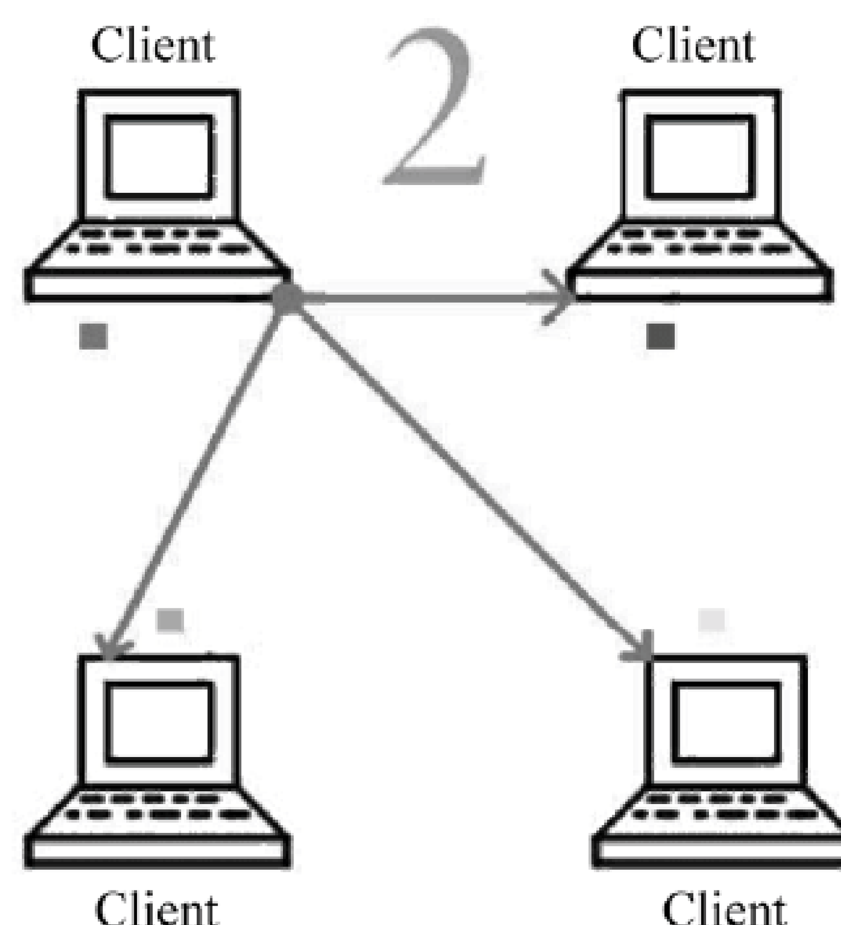


图 6.7　红色文件分片的交互

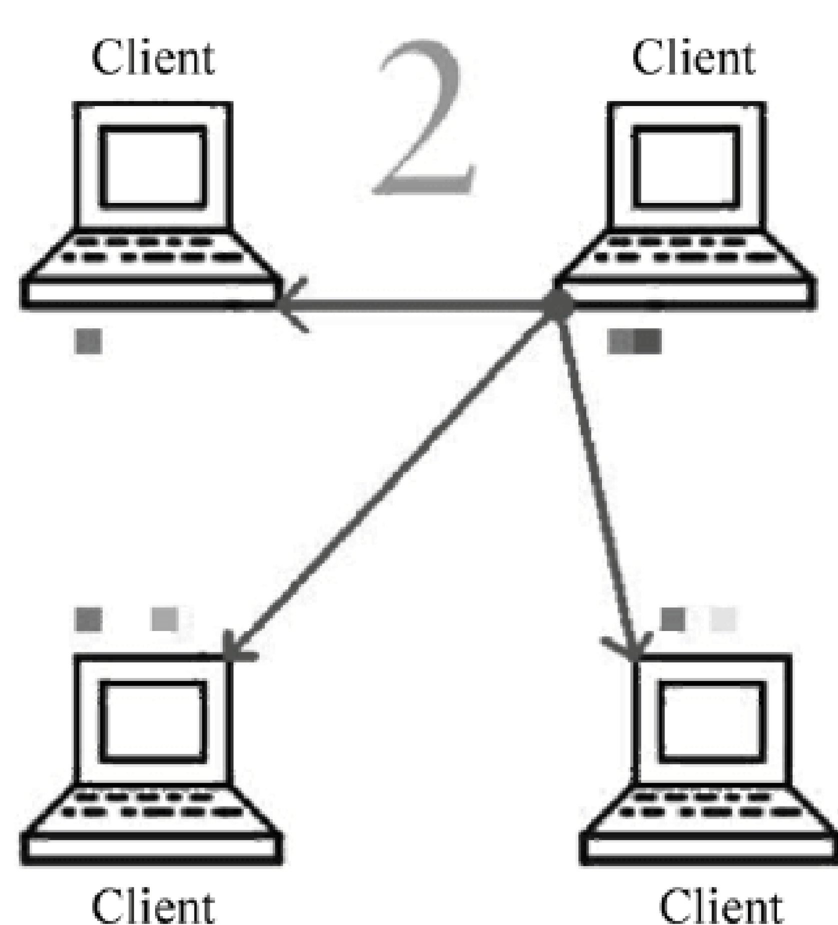


图 6.8　蓝色文件分片的交互

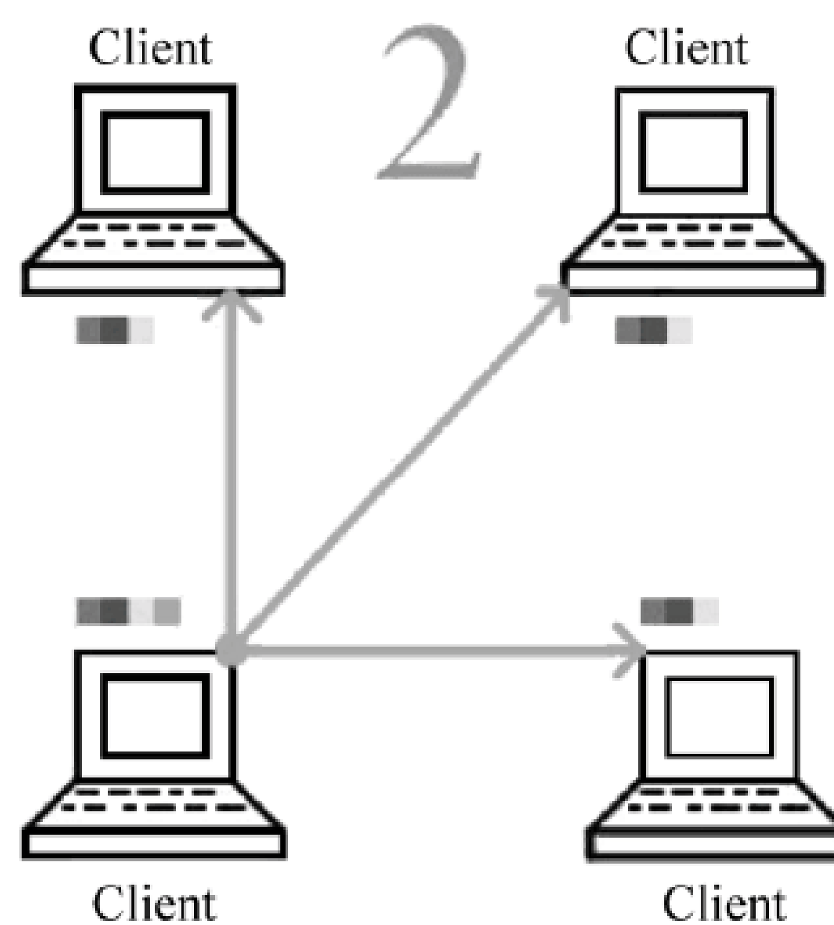


图 6.9　绿色文件分片的交互

- 拥有红色分片的 Client，通过 Client 之间的交互，把自己的分片文件，分别传给其他的 3 个 Client。
- 拥有蓝色分片的 Client，通过 Client 之间的交互，把自己的分片文件，分别传给其他的 3 个 Client。此时，其他的每个 Client 除了拥有自己的分片外，也拥有了红色

的文件分片。

- 拥有绿色分片的 Client，通过 Client 之间的交互，把自己的分片文件，分别传给其他的 3 个 Client。此时，其他的每个 Client 除了拥有自己的分片外，经过以前的两次交互也同时拥有了红色和蓝色的文件分片。
- 拥有黄色分片的 Client，通过 Client 之间的交互，把自己的分片文件，分别传给其他的 3 个 Client。此时，其他的每个 Client 除了拥有自己的分片外，经过以前的两次交互也同时拥有了红色、蓝色和绿色的文件分片。完成了 6.10 所示的交互之后，每个 Client 都拥有了 4 个全部的文件分片，经过验证组合，就会形成一个完整的文件。

注意：图 6.7～图 6.10 所示的交互过程是同时进行的，为了详细地说明下载的流程才将其分开描述，由图中的编号可知，这 4 个过程其实是在步骤（2）中同时完成的。

（3）Client 之间的交互完成，每个 Client 都拥有了一个完整的文件，如图 6.11 所示。

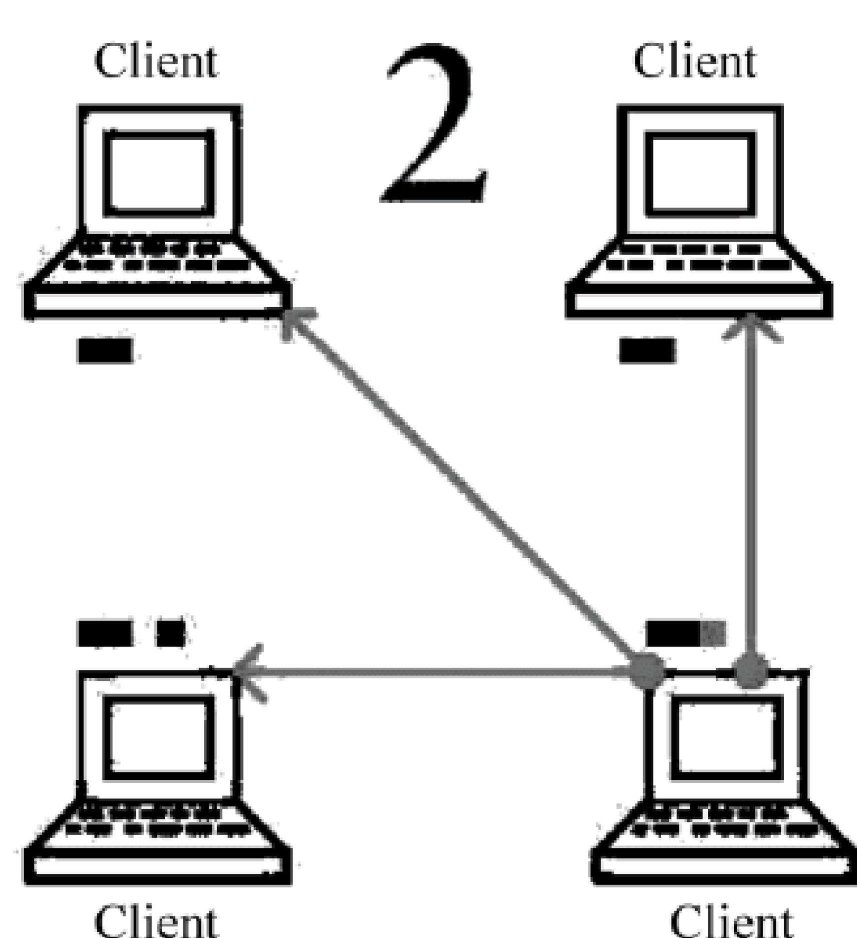


图 6.10　黄色文件分片的交互

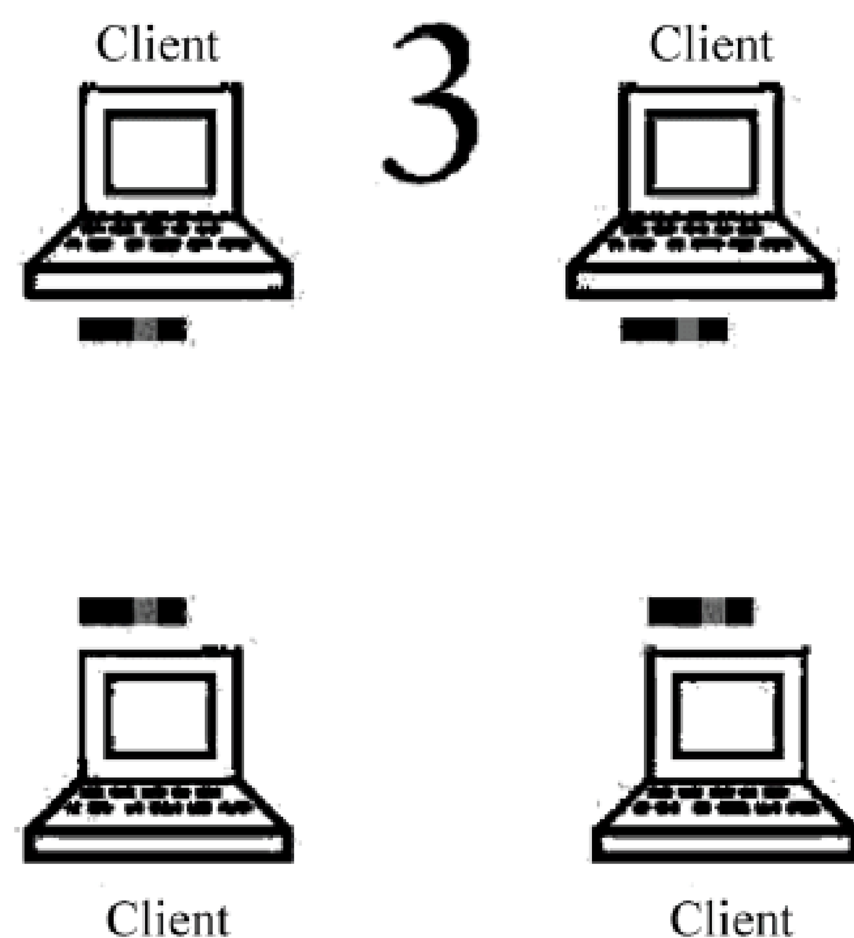


图 6.11　Client 之间完成交互文件下载结束

从整个下载过程中可以看出，文件下载的核心都集中在步骤(2)中，也就是 Peer（Client）之间资源的交互过程，这是 BT 下载流程的要义所在，也是其核心思想的体现。

2. 片段选择

通过 BT 下载的流程可知，在下载过程中将文件分片是个必须的过程，BT 将文件切割为固定大小的片段（典型的大小是 256K）。Peers 只有在检查了片段的完整性之后，才会通知其他 peers 拥有这个片段。在 BT 下载过程中，选择一个好的顺序来下载片段，对提高性能非常重要。一个差的片段选择算法可能导致所有的片段都处于下载中，或者另一种情况，没有任何片段被上传给其他 peers。

（1）严格的优先级

片段选择的第一个策略是，一旦请求了某个片段的子片段，那么该片段剩下的子片段优先被请求。这样，可以尽可能快地获得一个完整的片段。

（2）最少优先

对一个下载者来说，在选择下一个被下载的片段时，通常选择的是它的 peers 们所拥有的最少的那个片段，也就是所谓的“最少优先”。这种技术，确保了每个 peer 都拥有其他的 peers 们最希望得到的那些片段，从而一旦有需要，上传就可以开始。

（3）随机的第一个片段

“最少优先”的一个例外是在下载刚开始的时候。此时，下载者没有任何片段可供上传，所以，需要尽快地获取一个完整的片段。因此，第一个片段是随机选择的，直到第一个片段下载完成，才切换到“最少优先”的策略。

（4）最后阶段模式

有时候，peers 可能从一个速率很慢的 peer 那里请求一个片段。为了防止这种情况，在最后阶段，peer 向所有的 peers 都发送某片段的子片段的请求。一旦某些子片段到了，那么就会向其他 peers 发送“取消”消息，取消对这些子片段的请求，以避免带宽的浪费。

3．请求的同步（并发）发送

在 BT 协议中，很重要的一点是同时发送多个请求，以避免单个请求的两个片段发送之间的延迟。BT 协议将每个片段又进一步分为子片段，每个子片段的大小一般是 16K，同时，它一直保持几个请求被同时发送。流水作业选择同时发送的请求数目的依据是能使得大多数连接变得饱和。

6.3　BT 协议规范及分析

在 BT 的官方网站上，有两篇技术文档，一个是《BT 协议规范》，另一个是《Incentives Build Robustness in BitTorrent》，BT 协议规范是 BT 技术的核心，是其得以存在的理论基础，而后一篇则主要讲的是 BT 的一些设计思想。要研究 BT 技术，这两份文档几乎是仅有的资料。而 BT 协议规范是进行 BT 基础研究和 BT 应用开发的重要理论基础。本节就重点讲解一下 BT 协议规范，并对 BT 协议的要点进行分析。

注意：本节讲述的 BT 协议规范，来源于 BT 官方网站，http: //www.bittorrent.com/，读者可访问此网站下载原版的英文 BitTorrent Protocol Specification。由于在翻译习惯、常用术语上的区别，文章所讲内容可能有与读者的理解不一致的情况，如有任何出入，以官方的原版英文 BT 协议规范为主。

6.3.1　BT 协议规范说明

BT 是由布拉姆•科恩设计的一个端对端（peer to peer）文件共享协议，此协议使多个 peers 通过不可信任的网络的文件传输变得更容易。BT 是一个开源的规范性文档，处在不断的更新和变更中，下面是对此协议规范的说明。

1．BT协议的来源及维护

BT 协议规范的原文出自 BT 官方网站：http: //www.BT.com/protocol.html，此网站发布了协议的原文和更新的记录，此协议文档使用清楚明确的措辞书写，目前已成为一个正式的规范。当然，在某些研究中只作参考，还需附加某些特殊的讨论说明。

BT 协议规范文档，以开源的形式由 BT 开发社区维护和使用，其中的内容仅代表当前

协议，如有任何更新和变动，以 http: //www.BT.com 网站公布的版本和内容为主。

2．BT协议规范的应用范围

本文档适用于 BT 协议规范的第 1 版（v1.0）。目前，这份文档应用于 torrent 文件结构规范、peer wire 协议规范和 Tracker HTTP/HTTPS 协议规范。如果某个协议有了新的修订，请到对应页面查看。

注意：在本文档翻译过程中，如果遇到没有对应标准翻译的术语，一律不予翻译，例如 torrent、peer、tracker 等。

3．BT协议的相关约定

为了简明、准确地表达信息，在 BT 协议文档中，使用了许多约定，这些约定对理解 BT 协议的内容和思想有重要作用。

peer 与客户端（client）：在本文档中，一个 peer 可以是任何参与下载的 BT 客户端。客户端也是一个 peer，尽管 BT 客户端运行在本地机器上，但对 BT 协议规范的读者可能会认为自己是连接了许多 peer 的客户端。

分片（piece）与分块（block）：在 BT 协议文档中，片是指在元信息文件（metainfo file）中描述的一部分已下载的数据，它可通过 SHA-1 哈希函数来验证。而块是指客户端向 peer 请求的一部分数据。两块或更多块组成一个完整的可以被验证的片。

注意：本文在译用 BT 协议原文时，peer 一般翻译成“端”，也就是客户端的意思，所以 P2P 应该翻译成端对端，这与 P2P 中点对点、对等的意思并不冲突。当然，目前 BT 也没有一个标准的统一译法，因此本文对 Peer 不作翻译，同时读者应该将 peer to peer 和数据链路层的点对点协议（也缩写为 p2p）区分开。

4．BT协议的工作过程

BT 协议主要包括 3 个部分，即.torrent 文件的格式、tracker HTTP/HTTPS 协议和 Peer wire 协议（使用 TCP）。其中 tracker HTTP/HTTPS 协议是 BT 客户机与 tracker 服务器之间的通信协议，Peer wire 协议是 BT 客户机之间的通信协议。

注意：本节后面部分，会有对 BT 协议这 3 个核心内容的详细分析。

使用 Ethereal 工具，跟踪 BT 客户端在 BT 系统中下载一个文件的过程，然后分析 BT 协议的交互流程，可得 BT 协议各组件的工作时序图，如图 6.12 所示。

注意：这个时序图只是从整体上描述了 BT 协议的一个工作流程，结合下文对 BT 协议的详细分析，就能进一步理解。

6.3.2　BT 协议中的相关概念

本节主要讲解常见的一些文件或属性，如.torrent 文件等。

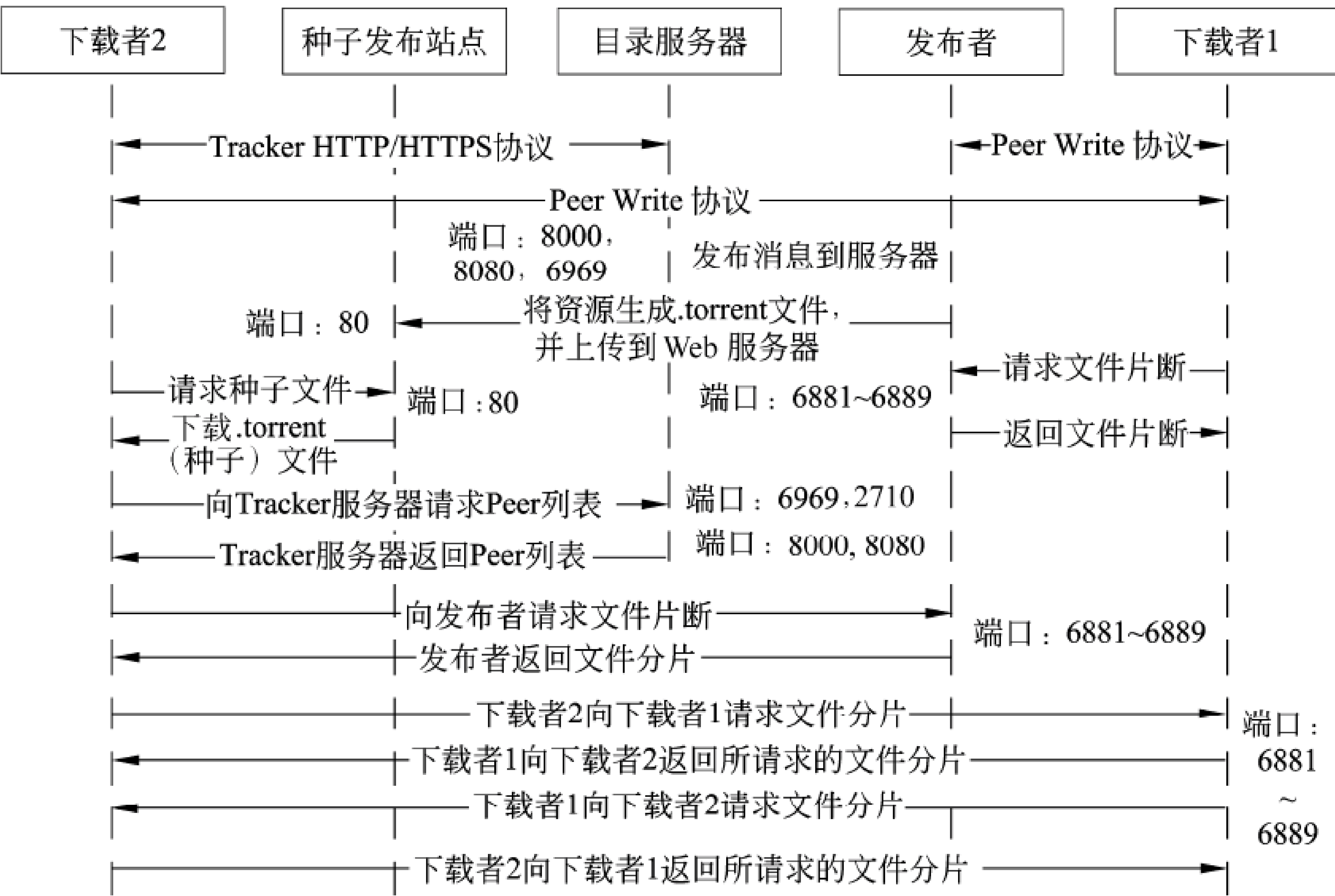


图 6.12　BT 工作的时序图

1．.torrent文件

.torrent 文件，扩展名为.torrent，包含了一些 BT 下载所必须的信息，这些信息主要有：

- ❑ 资源的名称，如果资源是目录形式，则还包括目录树中每个文件的路径信息和文件名。
- ❑ 如果资源是单个文件，则包括这个文件的大小信息；如果是以目录形式，则包括目录树中每个文件的大小。
- ❑ 对资源实际文件按照固定大小进行分块后，每块进行 SHA1 hash 运算得到的若干特征值的集合。
- ❑ .torrent 文件的创建时间、制作者填写的注释及制作者的信息等。
- ❑ 至少一个 announce 地址，对应于 Internet 上部署的一个 Tracker 服务器。

注意：关于元信息文件的结构在后文会有详细的讲解。

2．Tracker

Tracker 是指运行于服务器上的一个服务程序，也称 Tracker 服务器。这个程序能够追踪到底有多少人同时在下载或上传同一个文件。

客户端连上 Tracker 服务器，就会获得一个正在下载和上传的用户信息列表（通常包括 IP 地址、端口、客户端 ID 等信息）。根据这些信息，BT 客户端会自动连上别的用户进行下载和上传。

3．Client（客户端）

Client，泛指运行在用户自己电脑上的支持 BT 协议的程序。Client（客户）与 Trackers

服务器通信，这样，其他的客户端才能下载到那些发布的文件。

4．Seed（种子）

BT 把提供完整文件档案的人称为种子（Seed）。某一个文件现在有多少种子是可以看到的，只要有一个种子就可以放心地下载，一定能接收完。当然，种子越多、客户越多的文件接收起来的速度也就越快。

5．Re-Seed（补种）

拥有文件的人发布文件之后一段时间，很有可能有人未下完这个文件，这时下完的人就可以 Re-seed 一下，帮助那些还没有下载的朋友补完。

6．Hash（哈希）

Hash 是指用一小段数据来标识容量很大的一段数据，以验证它的完整性。在 BT 下载中，Hash 主要用来验证文件的完整性，并且 Hash 还可以作为不同文件判别的标志。

7．SHA1 hashing

SHA1 hashing 是 BT 使用的 hash 方式。

8．Announce

让全世界知道你已经发布文件了，别人可以来下载了。

9．UDP

UDP 协议是英文 UserDatagramProtocol 的缩写，即用户数据报协议，主要用来支持那些需要在计算机之间传输数据的网络应用。包括网络视频会议系统在内的众多客户/服务器模式的网络应用，都需要使用 UDP 协议。UDP 协议从问世至今已经被使用了很多年，虽然其最初的光彩已经被一些类似协议所掩盖，但是即使是在今天，UDP 仍然不失为一项非常实用和可行的网络传输层协议。

10．Re-release

对于一个发布很久，已经没有 Seed 的文件来说，可以重新制作.torrent 文件，然后提供下载。

11．Python

Python 是用来写 BT 软件的编程语言，BT 的创始人开发第一个 BT 系统时，就是用 Python 语言写成的。

6.3.3　BT 协议分析之——B 编码（Bencoding）

元信息文件和 tracker 的响应都采用的是一种简单、有效、可扩展的格式，被称为 Bencoding 编码的格式文件，也就是通常所说的 B 编码，它是一种以简洁的格式描述和组

织数据的方法。支持字节串、整数、lists 和 dictionaries 类型。由于对不需要的字典关键字可以忽略，所以这种格式具有可扩展性，其他选项可以根据这种格式定义方便地加进来。

根据 BT 协议规范的规定，B 编码支持 4 种类型的数据，它们分别是字节串类型（byte strings）、整数类型（integers）、列表类型（lists）、字典数据型（dictionarys）。下面分别对这 4 种类型的数据格式、定义、示例等进行讲解。

1. 字节串类型（byte strings）

对于字节串类型的数据，它的构建方式，首先是一个字符串的长度，然后是冒号，后面跟着实际的字符串。字节串的编码公式是：<以十进制 ASCII 编码的串长度>：<串数据>。

注意：字节串编码没有开始和结束分隔符。

例如，4：spam 表示的字节串，就是 spam 字符。在这个表示法中，4 表示的是这个字节串的长度，从冒号开始，读取 4 个长度的字符，那么 spam 就是它所表示的值。

2. 整数类型（integers）

整数类型的编码以字母 i 开始，然后是 10 进制的整数值，最后以 e 结尾。它的编码公式是：i<以十进制 ASCII 编码的整数>e。

例如，i3e 表示 3，i-3e 表示–3。整数没有大小限制，而 i-0e 是无效的，0 的 B 编码数值只能用 i0e 表示。除了 i0e 外，以 0 起始的整数都是无效的，例如 i05e，就是无效的。

注意：对于这个整数的最大位数规范并没有做出规定。

3. 列表类型（lists）

List，就是列表的意思，列表类型的 B 编码以 l 开始，接下来是列表值的编码（也采用 bencoded 编码，也就是嵌套的 B 编码），最后以 e 结束。它的编码公式是：l<B 编码值>e。

注意：开始的 l（l 是小写的 L，而不是大写的 i）与结尾的 e 分别是开始和结束分隔符。lists 可以包含任何 B 编码的类型，包括整数、串、dictionaries 和其他的 lists。

例如，l4：spam4：eggse 表示含有两个串的 lists，它的值就是：[“spam”、“eggs”] 这样的一个列表。

4. 字典数据型（dictionarys）

Dictionaries 就是字典编码，字典数据类型的 B 编码以 d 开始，接下来是可选的 keys 和它对应的值，最后以 e 结束。它的编码公式是：d<B 编码串><B 编码元素>e。

Dictionaries 编码中，开始的 d 与结尾的 e 分别是开始和结束分隔符。键（key）必须被 B 编码为串，并且以排序的顺序出现（以原始串排列，而不是以字母数字顺序）。串，采用二进制比较方式，而不是特定于某种文化的自然比较（既不是按照中文的比较方式，也不是按照英文的排序方式）。值（value）可以是任何 B 编码的类型，包括整数、串、lists 和其他的 dictionaries。

例如，d3：cow3：moo4：spam4：eggse，表示{‘cow’：‘moo’，‘spam’：‘eggs’}，而 d4：spaml1：al：bee 表示{‘spam’：[‘a’，‘b’]}。键值必须是字符串，而且已经排序（并非是按照字母顺序排序，而是根据原始的字符串进行排序）。

例，如下的一个 Dictionaries 编码表示的意思是什么？

d9：publisher3：bob17：publisher-webpage15：www.example.com18：publisher.location4：homee。

在这个 B 编码中，以 d 开头，以 e 结束，所以，整体而言一定是一个字典类型的编码。再看其内部的数据，都是字节串类型的编码数据，直接按字节串的编码规则将其解析出来即可。这样就可以根据 B 编码规则，将一个完整的字典编码清晰地解析出来。

结果表示：dictionary{"publisher"=>"bob","publisher-webpage" => "www.example.com","publisher.location" => "home"}。

注意：对于 string 和 integer，目前已经存在官方的翻译，但是 list 和 dictionary 并没有存在一个统一的译法，在本文中只是根据字面意思来直接翻译，如有不当之处以官方最新版本的解释为主。另外，本文中所有出现的例子，均来自于 BT 协议规范中的示例。

6.3.4　BT 协议分析之——元信息文件结构

元信息文件，就是 BT 下载中所说的.torrent 文件，也就是种子文件。它是按照 BT 协议规范编码规则进行编码后的文本文件，里面的所有数据都以 B 编码方式进行编码的，B 编码的规则在上文中已经有了详细的讲解。一个.torrent 文件的部分内容如图 6.13 所示。

```
d8:announce36:http://btfans.3322.org:8000/announce10:created by13:BitComet/0.7013:creation
datei1254147144e8:encoding3:GBK4:infod5:filesld6:lengthi332048e4:pathl35:Ice Age Dawn of t
Dinosaurs.a.jpge10:path.utf-8135:Ice Age Dawn of the Dinosaurs.a.jpgeed6:lengthi792875e4:p
:Ice Age Dawn of the Dinosaurs.jpge10:path.utf-8133:Ice Age Dawn of the Dinosaurs.jpgeed6:
i7970e4:pathl33:Ice Age Dawn of the Dinosaurs.txte10:path.utf-8133:Ice Age Dawn of the Din
eed6:lengthi1273016032e4:pathl96:影视帝国(bbs.cnxp.com).冰河世纪3： 大威龙驾到.Ice.Age.Dawn
Dinosaurs.2009.BluRay.720p.rmvbe10:path.utf-81110:褰辫   富濆浗(bbs.cnxp.com).鍐版渤涓栫邯3
.Ice.Age.Dawn.of.the.Dinosaurs.2009.BluRay.720p.rmvbeee4:name67:[2009.09.28]冰河世纪3： 大
国动画][720p]（帝国出品）10:name.utf-888:[2009.09.28]鍐版渤涓栫邯3锛氬ぇ濞佷緳椹惧埌[2009
[720p]锛堝笣鍥藉嚭鍝佹級12:piece lengthi524288e6:pieces48620:述?@竺V绡蠷舥D?□□亇□濬[   昢?f
```

图 6.13　.torrent 文件内容片断截图

注意：从网络上下载任意一个正确的.torrent 文件（种子文件），用普通的文本阅读器打开即可看到，图 6.13 就是 torrent 文件的内容。

根据 BT 协议规范，一个元信息文件的内容就是一个 B 编码的 dictionary，它包含下面列出的键（key），其中字符串类型的值均以 UTF-8 编码。

1．元信息文件的键值说明

元信息文件就是.torrent 文件，此文件是由满足 BT 协议的各个键所描述的，这些键的定义及说明如下。

- ❑ Info：该键（key）对应的值是一个描述.torrent 文件的 dictionary。整个 Info 的值是

一个 dictionary 类型的数据编码。该 dictionary 有两种可能的结构，一种对应于没有目录结构的单个文件的.torrent，另一种则对应于多文件的.torrent，如图 6.14 和图 6.15 所示。

注意：单文件的.torrent 就是种子文件中只包含一个文件，简单地说，我们在用 BT 下载时，打开.torrent 的种子文件时，只能下载一个文件。而多文件的.torrent，就是种子文件中包含多个文件。图 6.14 所示的，就是打开一个.torrent 的种子文件时，只有一个文件；而图 6.15 所示的，同样打开一个.torrent 的种子文件时，却含有多个文件。那么前者的种子就是单文件的.torrent，而后者的种子就是多文件torrent。



图 6.14　打开单文件.torrent 种子时的截图

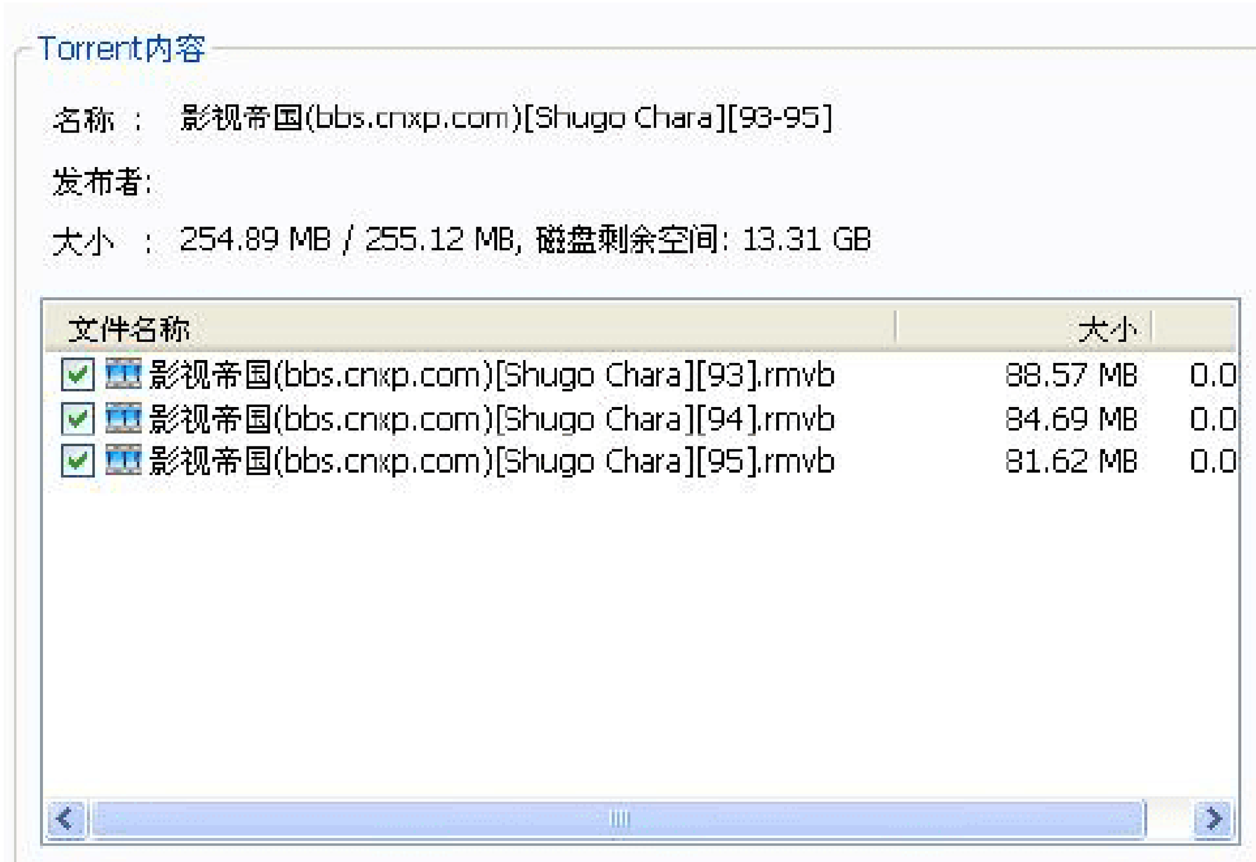


图 6.15　打开多文件.torrent 种子时的截图

- announce：该键对应的值是 tracker 的 announce URL，就是 tracker 服务器的地址值，这个值用一个字符串类型的数据表示（string）。
- announce-list：（可选），这是对正式规范的一个扩展，目的是提供向后兼容性。表示此键的数值，在 B 编码中是一个用字符串表示的列表型数据（list of lists of strings）。

注意：标注（可选）标记的，说明此键在元信息文件的结构构成中是可选的，也就是说，可以有此键，也可以没有此键。

- creation date：（可选），该键对应的值是 torrent 文件的创建时间，时间使用标准的 UNIX 时间格式。（整数类型，是从 1970-01-01 00：00：00 开始的秒数）。

注意：标准的 UNIX 时间格式指的是：UNIX 时间也叫做时间戳，保存的是格林威治标准时间从 1970 年 1 月 1 日零点起到当前时刻的秒数，以 32 位整列表示，其中 1970 年 1 月 1 日零点也叫 UNIX 纪元。

- comment：（可选），该键对应的值是.torrent 文件制作者的评论信息，用字符串类型的值表示。
- create by：（可选），该键对应的值是制作.torrent 文件的程序名称和版本信息，用符串类型的值表示。
- encoding：（可选），由上文可知，.torrent 元文件中包含一个 info dictionary。当

该 dictionary 过大时，就需要对其分片（piece），该编码就是用来生成分片的。该键值用字符串类型的数据表示。

以上对构成元信息文件的各个键值进行了说明，下面对键所对应值的详细情况进行说明。

2．Info Dictionary——Info键对应的值

在.torrent 文件中，在 info 标识的键值里记录了文件的诸多信息，在不同模式下对应着不同的值，它主要有两种模式，分别为单文件模式和多文件模式。不论在单文件模式还是在多文件模式下，有一些共有的键，这些键如下。

- piece length：该键对应的值是每一片（piece）的字节数。用整数类型表示。
- Pieces：该键对应的值由所有的 20 字节 SHA1 散列值连接而成的字符串，每一片（piece）均含有一个唯一的 SHA1 散列值。用字节串类型数据表示。
- private：（可选），这个键（key）所对应值是整数类型，如果设置为 1。客户端必须广播自己的存在，然后通过在元信息文件中显式描述的 trackers 得到其他的 peers。如果设置为 0 或者不设置，则表明客户端可以通过其他的方式来得到其他的 peers。例如 PEX peer exchange 技术，DHT 技术等。private 可以解释为没有外部的 peer 源（如果客户端不提供 PEX peer exchange 技术、DHT 技术等，那么 BitTorrent 客户端必须通过 tracker 来得到其他的 peers）。值用整数类型的数据表示。

注意： PEX peer exchange 技术是一种 P2P 网络中的来源交换技术，指的是从其他客户端之间进行来源交换，这样就能扩大 Peer 之间交互的深度和范围。在 BT 技术中，μTorrent 系统对 peer exchange 技术有着较好的支持。此技术也是当前 P2P 研究的一个热点。

除了以上公有的健以外，对单文件和多文件不同的模式而言，对应着不同的键。

Info in Single File Mode（单文件模式下的 Info 键）

在单文件模式下，info dictionary 包含如下结构。

- name：文件名。建议使用。值用字符串类型表示。
- length：文件的所占字节数。值用整数类型表示。
- md5sum：（可选），相当于文件 MD5 和 32 个字符的十六进制串，BT 根本就不使用这个键（key），但是有些程序为了更大的兼容性而包含它。值用字符串类型表示。

Info in Multiple File Mode（多文件模式下的 Info 键）

在多文件模式下，info dictionary 包含如下结构。

- name：存储所有文件的目录名，建议使用。值用字符串类型表示。
- files：由 dictionaries 的列表组成，多文件中的每一个文件都对应一个 dictionary，这些 dictionary 都存储在一个 list 中，而 list 下的每一个 dictionary 包含着 length、md5sum、path 的结构，具体描述如下：
  - length：文件的所占字节数。值用整数类型表示。
  - md5sum：（可选），相当于文件 MD5 和 32 个字符的十六进制串，BT 根本就不使用这个键（key），但是有些程序为了更大的兼容性而包含它。值用字符串

类型表示。

- path：包含单个或多个元素的 list，这些元素合成在一起表示文件路径和文件名。list 中的每一个元素对应于一个目录名或者文件名（当是最后一个元素时对应文件名）。

例如：文件“dir1/dir2/file.ext”由 3 个字符串元素组成："dir1"、"dir2"和"file.ext"。这 3 个元素会被编码成 B 编码的字符串 list：l4：dir14：dir28：file.exte。

注意：从多文件的键值结构中可以看出，其 list 中的 dictionary 字段刚好对应单文件模式下键（key），说得通俗点就是多文件模式是多个单文件模式的集合。

3．BitTorrent协议规范中对分片（piece）的说明

键（key），piece length 指出了片（piece）的正常大小，通常情况下 2 的 n 次方。一般根据 torrent 中文件数据的总大小来选择片大小，同时片太大导致低效，片太小，会使.torrent 文件太大，这两个因素都会影响片大小的选择。事实上，选择的片大小，应该使.torrent 文件不超过 50～75KB。

注意：之所以这样选择，将.torrent 文件的大小限定在一定的范围内，据推测这样能减轻存储.torrent 文件的 tracker 的负载。

目前最好的做法是保持片大小为 512KB 或者更少，虽然对于 8～10GB 的文件，会使.torrent 文件过大，但是片数量的增加有利于文件的共享。最常用的片（piece）大小是 256KB、512KB 和 1MB。

除了最后一块，所有的块都具有同样的大小，最后一块的大小是不规则的。这样片的数量就由（total length/piece size）决定。

对于多文件模式情况下的片边界，可以将文件数据当作是一个长的连续流，这个流由文件列表中的文件串连而成。这样多文件模式下片数量的决定方式和单文件模式下就一样了。片有可能跨越文件边界。

每个片都含有一个对应于该片数据的 SHA-1 散列值。这些散列值串连起来就形成了上述 info dictionary 中 pieces 键所对应的值（key-value）。需要注意的是这个值并不是一个由字符串组成的 list，而是一个字符串，字符串的长度是 20 的倍数。

6.3.5　Bencoding 编码与解码的编程实现

上文已经对元信息文件（.torrent）的结构，及 B 编码的规则进行了详细的讲解。那么根据这一结构就可以构造算法，对任意文件进行编码形成.torrent 文件，也可以对任意的.torrent 文件进行解码，解析出其中的信息。下面以编程的方式来实现.torrent 文件的编码与解码。

B 编码，是一个比较简单的算法实现，有很多种实现方式，当前也有很多开源的源程序代码。BT 原作者是用 Python 实现的，其他的编程语言如 C、Perl、Java、Objective-c 等都可以实现。本文就用在 Windows 和 Linux 平台下均适用的 Java 语言来实现 B 编码程序。

注意：下文只对程序中用到的主要方法进行说明，详细的程序源代码请读者自行参考随书光盘所附的源代码。

【BEncoder.java 类示例参考：\源代码\ch6\ch6_code\bencode\BEncoder.java】

BEncoder.java 类，主要是根据 B 编码规范，对数据流进行 Bencoding 编码。在实际应用中，就是将一个文件的信息制作成 Torrent 文件的过程。详细的代码实现请参考随书光盘中的源代码部分，这里只简要地说明几个主要的方法。

```
/*
 * BEncoder，B 编码的编码类，根据 B 编码的规则，主要用于对一个文件进行 B 编码
 */
import java.io.*;                          //引入执行相关的 Java.io 包
import java.util.*;                        //引入执行相关的 Java.util 包
//编码方法，将一个对象编码成字节数据
public class BEncoder {
    //bencode()方法，传入一个对象类型，按照 B 编码规则，将不同的对象编码成不同的格式
    public static byte[] bencode(Object o) throws IllegalArgumentException
{
        try {
            //定义一个字节输出流
            ByteArrayOutputStream baos = new ByteArrayOutputStream();
            //调用另一个编码方法，需要传入一个编码对象、字节流作为参数
            bencode(o, baos);
            //返回一个字节数组
            return baos.toByteArray();
        } catch (IOException ioe) {           //捕获 I/O 异常
            throw new InternalError(ioe.toString());
        }
}
```

以下的方法，是另一个 bencode 编码方法，接收一个编码对象和一个输出流作为参数，主要作用是对传入的对象进行判断。如果判定对象类型是 String 类型，则调用 String 类型的编码方法，如果是整数类型则调用整数类型的编码方法……，不同的数据类型其编码方式是不一样的。

```
//这是上文中调用的另一个编码方法，编码对象和字节形式的输出流作为参数
public static void bencode(Object o, OutputStream out) throws IOException,
            IllegalArgumentException {
//利用 Java 中的 instanceof 运算符对传入的不同数据类型进行判断。根据不同的类型，调用不同的编码方法
        if (o instanceof String)       bencode((String) o, out);
                                                    //处理字符串类型的数据
        else if (o instanceof byte[])  bencode((byte[]) o, out);
                                                    //处理字节数组类型的数据
        else if (o instanceof Number)  bencode((Number) o, out);
                                                    //处理整数类型的数据
        else if (o instanceof List)    bencode((List) o, out);
                                                    //处理列表类型的数据
        else if (o instanceof Map)     bencode((Map) o, out);
                                                    //处理字典类型的数据
        else if (o instanceof BEValue) bencode((BEValue) o, out);
                                                    //处理编码数据的值
        else
        //如果编码过程中出现了规范以外的其他数据类型，则抛出不合法的参数异常
```

```
            throw new IllegalArgumentException("Cannot bencode: " + o.
           getClass());
}
//以下的方法是对传入的 BEValue 值对象进行处理，具体可参考 BEValue 类
    public static void bencode(BEValue value, OutputStream out) throws
    IOException {
     Object o = value.getObject();       //调用 BEValue 类的 getObject()方法
         if (o instanceof Map)            //判断此对象是否是 Map 类型
             bencode((Map) o, out);       //调用 Map 类型数据的编码方法
         if (o instanceof List)           //判断此对象是否列表类型
             bencode((List) o, out);      //调用列表类型数据的编码方法
         if (o instanceof Number)         //判断此对象是否整数类型
             bencode((Number) o, out);    //调用整数类型数据的编码方法
         if (o instanceof byte[])         //判断此对象是否 Byte 类型
             bencode((byte[]) o, out);    //调用 Byte 类型数据的编码方法
}
```

以下的方法是对字符串类型的数据进行编码的过程，该方法接收用户输入的字条串，根据编码规则，读取字符串数据流，返回一个编码后的字节数组。

```
    //对字符串数据进行编码
    public static byte[] bencode(String s) {
        try {
            //定义一个字节数据输出流对象
            ByteArrayOutputStream baos = new ByteArrayOutputStream();
            //调用字符串数据的编码方法
            bencode(s, baos);
            //返回一个字节数组
            return baos.toByteArray();
        } catch (IOException ioe) {
            throw new InternalError(ioe.toString());
        }
}
//对字符串数据进行编码的具体实现，需要传入字符串内容和输出流对象作为参数
    public static void bencode(String s, OutputStream out) throws IOException
{
        byte[] bs = s.getBytes("UTF-8");
        bencode(bs, out);
}
```

以下方法是对整型数据进行编码，需要传入一个整数的数据值作为参数，在方法体中读取整型数据，经处理后返回一个字节数组。

```
    //对整型数据进行编码的过程
    public static byte[] bencode(Number n) {
        try {
            //定义一个字节输出流对象
            ByteArrayOutputStream baos = new ByteArrayOutputStream();
            //调用整型数据的编码方法
            bencode(n, baos);
            //返回编码后的字节数组
            return baos.toByteArray();
        } catch (IOException ioe) {
            throw new InternalError(ioe.toString());
        }
}
//对整型数据进行 B 编码的具体实现
```

```
public static void bencode(Number n, OutputStream out) throws IOException
{
//根据 B 编码规则，整型数据在元信息文件中是以字符 i 开头的，所以在输出流中先写入字符 i，表示开始对整型数据进行编码
        out.write('i');
        String s = n.toString();          //将整型数据转换为字符串
        out.write(s.getBytes("UTF-8")); //将字符串以 UTF-8 的编码形式写出
        out.write('e');                   //编码完成后，写入 e 字符作为结束标记
}
```

以下方法是对列表（List）型数据进行编码的实现，需要传入一个 List 对象作为参数，在方法体中读入列表型的数据对象，返回一个编码后的字节数组。

```
    //对列表类型的数据进行编码
    public static byte[] bencode(List l) {
        try {
            ByteArrayOutputStream baos = new ByteArrayOutputStream();
            //调用编码方法
            bencode(l, baos);
            return baos.toByteArray();
        } catch (IOException ioe) {
            throw new InternalError(ioe.toString());
        }
}
//对列表型数据进行编码的具体实现
public static void bencode(List l, OutputStream out) throws IOException {
//根据 B 编码规则，列表型数据在元信息文件中是以字符 l 开头的，所以在输出流中先写入字符 l，表示开始对列表型数据进行编码
        out.write('l');
        //对列表中的所有数据进行遍历，并对其中的每一个数据都进行编码
        Iterator it = l.iterator();
        while (it.hasNext())
            bencode(it.next(), out);      //调用另一个编码方法
        out.write('e');                   //编码完成后，写入 e 字符，表示编码结束
}
//参数为字节数据的编码方法，用来对字节数组进行编码
    public static byte[] bencode(byte[] bs) {
        try {
            ByteArrayOutputStream baos = new ByteArrayOutputStream();
            bencode(bs, baos);
            return baos.toByteArray();
        } catch (IOException ioe) {
            throw new InternalError(ioe.toString());
        }
}
//参数为字节数组和输出流对象的编码方法
    public static void bencode(byte[] bs, OutputStream out) throws
IOException {
        String l = Integer.toString(bs.length);     //取得字节的长度
        out.write(l.getBytes("UTF-8"));             //转换为 UTF-8 编码
        out.write(':');                             //写入一个冒号
        out.write(bs);                              //写入列表中的值
}
```

以下方法是对字典类型的数据进行编码，整个.torrent 元信息文件就是一个字典型的结构，包括了多个嵌套的字典类型结构。在方法的具体实现过程中，需要输入一个字典数据类型的数据作为参数，方法体中调用字典类型数据编码的具体实现方法，返回一个编码后

的字节数组。

```
    //对字典类型的数据进行编码，在具体的实现中，用 Map 的结构来存储字典型数据
    public static byte[] bencode(Map m) {
        try {
            ByteArrayOutputStream baos = new ByteArrayOutputStream();
            //调用编码方法的具体实现
            bencode(m, baos);
            //返回编码后的结果
            return baos.toByteArray();
        } catch (IOException ioe) {
            throw new InternalError(ioe.toString());
        }
}
//字典型数据编码的具体实现，需要传入一个 Map 类型数据和一个输出流对象
public static void bencode(Map m, OutputStream out) throws IOException {
    //首先需要写入字典类型数据编码的开始标记符 d
        out.write('d');
        //根据 Map 结构中的 keySet()方法，取出所有的 Key 值，放到一个 Set 集合里
        Set s = m.keySet();
        //将 Set 集合中所有的 Key 值存储到一个 List 中
        List l = new ArrayList(s);
        //对 List 中所有的 Key 值进行排序
        Collections.sort(l);
        //用 Iterator()方法，遍历 List 中的所有的 Key 值
        Iterator it = l.iterator();
        while (it.hasNext()) {
            //将 Key 强制转换成 String 类型
            String key = (String) it.next();
            //根据 Key 值，从 Map 结构中取出每一个 Key 对应的 Value 值
            Object value = m.get(key);
            //对每一个 Key 值进行编码
            bencode(key, out);
            //对每一个 Key 所对应的 Value 值也进行编码
            bencode(value, out);
        }
        //编码完成后，写入字典类型的编码结束标记符 e
        out.write('e');
    }
}
```

以上是对.torrent 元信息文件按 B 编码算法进行编码的主要实现方法。这里只对一些主要方法所体现的 B 编码算法思想、实现思路进行了简要的说明，完整的可运行程序请读者自行参考相应的源代码。

.torrent 元信息文件是经 B 编码进行编码后得到的文件，要抽取出元信息文件中的数据信息，还必须要能对其进行正确的解码。下面就讲解一下，如何对一个 B 编码的文件进行解码。

本文中，具体的解码类由 BDecoder.java 类来实现，解码的过程其实就是编码的逆过程，解码过程的算法依据也是 B 编码规范，最终目的是要将一个.torrent 元信息文件中所包含的信息全部抽取出来。

在程序实现的过程中，根据 BT 的编码规范，逐个扫描.torrent 文件内容，分别根据字符串、整数、列表等编码特征读取不同的值到相应的数据结构中，再解析此数据结构，从中抽取出原始的值。

【BDecoder.java 类示例参考：\源代码\ch6\ch6_code\bencode\BDecoder.java】

在此程序的设计中，要十分注意编码的规则、编码方式、完整性、正确性等要求。下面就是 BDecoder.java 类中几个主要方法的简要说明，详细的实现请参考相应的源代码。

```
/*
* BDecoder，B 编码的解码类，根据 B 编码的规则，主要将一个 B 编码的数据解析出来
*/
import java.io.*;
import java.math.BigInteger;
import java.util.*;
import java.security.*;
public class BDecoder {                          //对 B 编码的规则定义
    private final InputStream in;                //定义一个私有的输入流
    private String special_map = "info"; //将整个 Info 信息放到一个 Map 的结构中
    private boolean in_special_map = false;
    private final MessageDigest sha_digest;
    /**
     * 构造方法，初始化 BDecoder，接收一个输入流，并进行 SHA1 校验
     * @param in
     */
    public BDecoder(InputStream in) {
        this.in = in;
        try {
            //利用 SHA1()函数得到消息摘要
            sha_digest = MessageDigest.getInstance("SHA");
        } catch (NoSuchAlgorithmException nsa) {
            throw new InternalError(nsa.toString());
        }
}
```

以下方法是对编码值的一个解码过程。根据编码规则，不同类型的数据有不同的编码方式，要解析这些数据，就是一个.torrent 文件编码的逆向过程，需要分别对编码过程中出现的特征字符如 i（表示整数的开始）、l（表示列表数据的开始）、d（表示字典型数据的开始）等进行判断，并从中抽取出值内容。实现过程如下：

```
    /**
     * Method name:bdecode，核心的解码过程
     * TODO : 对一个流式的文件进行 B 编码的解码操作
     */
    public static BEValue bdecode(InputStream in) throws IOException {
        return new BDecoder(in).bdecode();
}
//以字节的方式，读取文件中的内容
    public int getNextIndicator() throws IOException {
        if (indicator == 0) {
            indicator = in.read();          //读取一个字节的数据，用 int 型表示
               if (in_special_map)          //判断是否是一个字典结构的开始
                sha_digest.update((byte) indicator);
        }
        return indicator;                   //返回读取的值
}
//对读取的每一个值进行解码处理，返回一个 BEvalue 结构
    public BEValue bdecode() throws IOException {
        indicator = getNextIndicator(); //取出从文件中读出的一个字节值
        if (indicator == -1)            //如果文件读取结束，直接返回 null
            return null;
```

```
        if (indicator >= '0' && indicator <= '9')
                                      //如果读取的是一个数字，直接进行字节解码
            return bdecodeBytes();        //返回字节解码的结果
        else if (indicator == 'i')        //如果读取的数据是字符 i 开头的
            return bdecodeNumber();       //对整数类型的数据进行解码
        else if (indicator == 'l')        //如果读取的数据是字符 l 开头的
            return bdecodeList();         //对列表类型的数据进行解码
        else if (indicator == 'd')        //如果读取的数据是字符 d 开头的
            return bdecodeMap();          //对字典类型的数据进行解码
        else                              //否则抛出相应的异常
      throw new InvalidBEncodingException("Unknown indicator '" + indicator
      + "'");
}
```

以下的方法主要用来处理字符串类型的数据编码，根据 B 编码的规则，字符串类型数据的编码方式是十进制 ASCII 编码的串长度：串数据。这样，在解析的时候根据这个规则，首先读取描述字符串长度的十进制数，然后根据这个数读取后续的字符串即可。

```
    /**
     * Method name:bdecodeBytes，返回一个 BEvalue 的值，抛出 IOException
     * TODO : 对字节串类型的数据进行解码
     */
    public BEValue bdecodeBytes() throws IOException {
        int c = getNextIndicator();       //从文件中读取一个字符
        int num = c - '0';     //定义一个 num 变量，取字符‘C’与‘0’的 ASCII 差值
        if (num < 0 || num > 9)           //对 num 的值进行判断
        ///如果超出范围，则抛出相应异常
        throw new InvalidBEncodingException("Number expected, not '" + (char)
        c + "'");
        //以下是解码过程
        indicator = 0;
        c = read();
        int i = c - '0';
        while (i >= 0 && i <= 9) {
            num = num * 10 + i;
            c = read();
            i = c - '0';
        }
       //根据编码规范，对冒号“：”进行判断
        if (c != ':')
        throw new InvalidBEncodingException("Colon expected, not '"+ (char)
        c + "'");
        return new BEValue(read(num));
}
```

以下的方法主要用来处理数据类型的数据编码。根据 B 编码的规则，整数数据类型在编码的时候有开始符也有结束符，编码以字母 i 开始，然后是 10 进制的整数值，最后以 e 结尾。这样，在解析的时候，如果遇到字母 i 说明以整数类型的数据编码开始，然后读取数据内容，直接遇到字母 e 为止。下面是此方法的具体实现。

```
    /**
     * Method name:bdecodeNumber，返回一个 BEvalue 的值，抛出 IOException
     * TODO : 对整型数据进行解码
     */
    public BEValue bdecodeNumber() throws IOException {
        int c = getNextIndicator();    //从文件中读取了一个整型表示的字符
```

```
        if (c != 'i')                //判断此字符是否整数型数据编码的开始标记 i
        //如果不是以 i 字符标记为开头，则抛出相应的异常
        throw new InvalidBEncodingException("Expected 'i', not '"+ (char) c
        + "'");
        indicator = 0;
        c = read();                  //读取文件
        if (c == '0') {
            c = read();
            if (c == 'e')            //判断编码是否结束
             return new BEValue(BigInteger.ZERO);
            else
         throw new InvalidBEncodingException("'e' expected after zero," +
         "not '" + (char) c + "'");
        }
        char[] chars = new char[256];    //定义一个字符数组
        int off = 0;                     //标记一个开始位置
        if (c == '-') {
            c = read();                  //读取内容并解码
            if (c == '0')
                throw new InvalidBEncodingException("Negative zero not
                allowed");
            chars[off] = (char) c;       //将读取的结果存入到新开辟的字符数组中
            off++;                       //标记向前推进一个位置
        }
        if (c != 'e')                    //对结束标记进行判断，判断编码是否正常结束
            throw new InvalidBEncodingException("Integer should end with
            'e'");
        String s = new String(chars, 0, off);
        return new BEValue(new BigInteger(s));
}
```

与上面所讲的方法类似，下面的方法主要用来处理 List 类型的数据编码。根据 B 编码规则，List 类型的数据，以 l 标记作为开始符，以 e 标记作为结束符。在扫描文件内容的时候，将这两个标记中的数据取出，就是 List 型数据的内容了。实现方法如下：

```
    /**
     * Method name:bdecodeList，返回一个 BEvalue 的值，抛出 IOException
     * TODO : 对 List 列表型数据进行解码
     */
    public BEValue bdecodeList() throws IOException {
        int c = getNextIndicator(); //从文件中取出下一个字符
        if (c != 'l')                //判断此字符是否 List 类型数据编码的开始标记
        //如果不是以 l 标记开头，则抛出异常信息
        throw new InvalidBEncodingException("Expected 'l', not '" + (char)
        c + "'");
        indicator = 0;               //定义一个指示器
        List result = new ArrayList();   //定义一个名为 result 的 List 对象
        c = getNextIndicator();          //取出下一个字符
        while (c != 'e') {               //判断是否读到结束标记
            result.add(bdecode());       //将解码后的内容存入到 result 中
            c = getNextIndicator();      //接着往下再取下一个字符
        }
        indicator = 0;
        return new BEValue(result);      //以 BEvalue 的形式返回结果
}
```

以下的方法主要用来处理字典类型的数据编码。字典，顾名思义就是跟字典类似的数

据结构。它有一个<key,value>对，key 是索引，value 是值，用 Map 结构来存储这类数据。以下是字典型数据解码的实现。

```
//用于解码字 B 编码中字典类型的数据，返回一个 BEvalue 类型的值
public BEValue bdecodeMap() throws IOException {
        int c = getNextIndicator();        //取出下一个字符
        if (c != 'd')          //判断是否是字典类型数据的开始标记，如果不是则抛出异常
       throw new InvalidBEncodingException("Expected 'd', not '" + (char) c
       + "'");
        indicator = 0;                     //定义一个指示器
        //定义 Map，用了 Java 里的泛型知识
        Map<String, BEValue> result = new HashMap<String, BEValue>();
        c = getNextIndicator();            //读出下一个字符
        while (c != 'e') {                 //是否是结束标记
            //取出字典类型数据的 Key 值，注意字典型数据的键值都是字符串类型的
            String key = bdecode().getString();
            //通过 special_map.的 equals 方法对 Key 值进行判断
            boolean special = special_map.equals(key);
            if (special)
                in_special_map = true;
            BEValue value = bdecode();
            ///将解码后的 Key、Value 值放到 result 的 Map 结构中
            result.put(key, value);
            if (special)
                in_special_map = false;
            //接着取下一个字符
            c = getNextIndicator();
        }
        indicator = 0;
        //返回 BEvalue 结构的解码结果
        return new BEValue(result);
    }
```

以上的几个方法只是在 BDecoder 类中抽取几个重要的方法进行说明，因为各个方法之间并不是孤立的，有很多辅助方法并没有在文中显示，读者需要结合源代码来学习并理解 B 编码文件的详细解码过程。

本章所列的程序代码在 Java 1.6、Windows XP、Linux 2.6 平台下验证通过。可以将这几段程序打成 Jar 包，在工程中直接调用。读者可以根据程序的接口规则，自行编写一个测试程序，以验证 B 编码规则下的编码、解码过程。

注意： 以上程序涉及的引用类、Jar 包等，可以在随书光盘第 6 章的内容中找到。还有用其他语言开发的类似这种 B 编码的编码、解码程序，有兴趣的读者也可自行开发。

6.3.6　BT 协议分析之——Tracker 的 HTTP/HTTPS 协议

纵观整个 BT 的协议规范，重点讲述了 3 个核心内容，一个是元文件信息结构，这在上文已经讲过，另一个是 Tracker 的 HTTP/HTTPS 通信协议，还有一个就是 Peer 端协议。要理解 BT 协议规范，这 3 个核心内容是需要重点掌握的，本节就讲一下 Tracker 的 HTTP/HTTPS 协议。

1. Tracker的Get请求

Tracker 是一个响应 HTTP GET 请求的 HTTP/HTTPS 服务。这个请求包含来自客户端的度量信息，这些信息能够帮助 Tracker 全面地统计.torrent 信息。

Tracker 的响应包含一个 peers 列表，这个列表能够帮助客户端加入到.torrent 中。Base URL 由元数据文件（即以.torrent 为后缀的文件）中定义的 announce URL 组成，然后使用标准的 CGI 方式将这些请求参数追加到这个 URL 后面。

注意：CGI 方式即在 announce URL 后面紧跟一个“?”，然后是一个以“&”分隔的 param=value 序列，类似于 HTTP 中 GET 请求的参数列表形式。

在 announce URL 中的所有二进制数据，特别是 info_hash 和 peer_id，必须适当地进行转义。这意味着不在集合{0-9，a-z，A-Z，‘。’，‘-’，‘_’，‘~’}中的字节必须以“%nn”方式编码，其中 nn 是这个字节的十六进制值。

例：对于一个 20 字节的散列值的 URL 表示方法

如：“\x12\x34\x56\x78\x9a\xbc\xde\xf1\x23\x45\x67\x89\xab\xcd\xef\x12\x34\x56\x78\x9a”。正确的编码应该是：“%124Vx%9A%BC%DE%F1%23Eg%89%AB%CD%EF%124Vx%9A”。

注意：以上这种转义的 URL 编码方式，请阅读 RFC1738 文档中关于 URL（统一资源定位符，Uniform Resource Locators）格式规定说明。可参考网址：http: //www.faqs.org/rfcs/rfc1738.html。

2. Tracker的请求参数（Tracker Request Parameters）

从客户端发送到 Tracker 的请求包含如下参数。

（1）info_hash：URL 编码的 20 字节 SHA1 散列，这个散列是元信息文件中 info 键所对应的值的 SHA1 散列。info 键所对应的值是一个 B 编码的 dictionary，关于 info 键的定义在上文中已有详细的讲解。

（2）peer_id：使用 URL 编码的 20 字节串，用于标识客户端的唯一 ID，由客户端启动时生成。这个 ID 可以是任意值，甚至可能是二进制数据。目前没有生成该 ID 的标准准则，尽管如此，在实现的时候必须保证该 ID 对于本地主机是唯一的，因此可以通过包含进程 ID 和启动时记录的时间戳（timestamp）来达到这个目的。

（3）port：客户端正在监听的端口号。为 BT 协议保留的端口号是 6881～6889。如果不能使用该区间的数字建立端口号，客户端有可能拒绝建立连接。这个端口对 NAT 穿越和端口映射也有重要的作用。

（4）uploaded：客户端已经上传的总量（从客户端发送 started 事件到 Tracker 算起），以十进制 ASCII 码表示。尽管官方规范没有显式指定，大部分情况下是指已经上传的字节总数。

（5）downloaded：是与 upload 对应的，表示已下载的字节总量（从客户端发送 started 事件到 Tracker 算起），以十进制 ASCII 码表示。尽管官方规范没有显式指定，大部分情况下是指已经下载的字节总数。

（6）left：客户端还没有下载的字节数，以十进制 ASCII 码表示。

（7）compact：如果设置为 1，表示客户端接收压缩的响应包。这时 peers 列表将被 peers 串代替，在这个 peers 串中，每个 peer 占 6 个字节。这 6 个字节的前 4 个字节表示主机信息（以大端表示即以网络字节序），后两个字节表示端口号（以大端表示即以网络字节序）。需要注意的是，为了节约带宽，有些 Tracker 只支持返回压缩的响应包，如果没有将 compact 设置为 1，Tracker 将拒绝这个请求，或者如果请求中不包括 compact=0 这个参数，Tracker 将返回压缩的响应包。

（8）no_peer_id：表示 Tracker 将省略 peers dictionary 中的 ID 字段。如果启用 compact，那么就会忽略这个选项。

（9）event：如果指定的话，必须是 started、completed、stopped 和空之中的一个。如果一个请求不指定 event，表明它只是每隔一定间隔发送的请求。Event 事情取值的意思如下：

- started：第一个发送到 Tracker 的请求，其 event 值必须是该值。
- stopped：如果正常关闭客户端，必须发送该事件到 Tracker。
- completed：如果下载完毕，必须发送该事件到 Tracker。如果客户端启动之前，已经下载完成的话，则没有必要发送该事件。Tracker 仅仅基于该事件增加已经完成的下载数。

注意：如果 event 指定了值，但值为空，那么它和 event 不指定值的效果是一样的，表示的是每隔一定间隔发送的请求。

（10）ip：（可选）客户主机的 IP 地址，点分十进制 IPv4 或者 RFC3513 指定的十六进制 IPv6 地址。

注意：一般情况下这个参数没有必要，因为 IP 地址可以从 HTTP 请求中得到。只有当请求中的 IP 地址不是客户端的 IP 地址时，才需要这个参数。也就是说，只有客户端通过代理的方式和 tracker 交互时，才会发生这种情况。

如果客户端和 Tracker 都在 NAT 网关的同一侧，这时候这个参数就是必要的。原因是 Tracker 会公布不能路由的客户端物理地址。因此客户端必须显式指出自己的 IP 地址，以将信息公布给外部的 peers。不同的 Tracker 对待该参数的方式不同。一些 Tracker 只有当请求中的 IP 地址在私有地址（RFC1918 规范）空间时才使用它，有一些则无条件使用，另一些则根本不使用。如果是 IPv6 地址，它表示这个客户端只能通过 IPv6 交互。

注意：在本文中涉及了很多 RFC 的文档规范，如 RFC3513 指的是 Internet Protocol Version6（IPv6）Addressing Archite，即 IPv6 的相关规范。而 RFC1918 指的是 Address Allocation for Private Internets，即私有网络地址分配的规范。这些规范文档读者在需要时请自行查阅。

（11）numwant：（可选）客户端希望从 Tracker 接受到的 peers 数，如果省略，则默认是 50 个。

（12）key：（可选）不和其他用户共享的附加标识。当客户端 IP 地址改变时，可以使用该标识来证明自己的身份。

（13）trackerid：（可选）如果之前的 announce 包含一个 tracker id，那么当前的请求必须设置该参数。

3．Tracker服务器的应答（Tracker Response）

Tracker 以“text/plain”文本响应客户端的请求，这个响应文本由 B 编码的 dictionary 组成，而这个 dictionary 则包含如下的键（key）。

- failure reason：如果 dictionary 中包含这个键（key），那么其他的键（key）就不会出现，该键（key）对应的值是一个可读的错误消息，它告诉用户的请求为什么会失败。值用字符串类型数据表示。
- warning message：（可选）类似于 failure reason，但是即使存在这个键，这个响应还会正常地处理下去。warning message 显示为一个错误，值用字符串类型的数据表示。
- interval：客户端每隔一定间隔就会向 Tracker 发送一个请求，这个键（key）以秒为单位指出这个间隔的大小。值用整数类型数据表示。
- min interval：（可选）最小的请求间隔，它表示客户端不能在这个时间间隔之内向 Tracker 重发请求。值用整数类型的数据表示。
- tracker id：客户端发送其下一个请求时必须返回给 Tracker 的一个字符串。如果缺失，但是上一个请求发送了 tracker id，那么不要丢弃旧值，重复利用即可。
- complete：完成整个文件下载的 peers 数，即做种者的数量。值用整数类型数据表示。
- incomplete：非做种的 peers 数（还没有完成该文件下载的 peers 数），即“占他人便宜者”数。值用整数类型的数据表示。
- peers：（dictionary model）该键（key）对应的值是一个 dictionary list（列表），该 list 中的每一个 dictionary 都包含如下的键（key）：
  - peer id：peer 自己选择的用来标识自己的 ID，上文在描述 Tracker 请求时已经说明。值用字符串类型数据表示。
  - ip：peer 的 IP 地址，可以是 IPv6/IPv4/DNS name。值用字符串类型数据表示。
  - port：peer 的端口号。值用整数类型数据表示。
- peers：（binary model）不是使用上面描述的 dictionary model，在 binary model 下，该键（key）对应的值可以有多个 6 字节组成的字符串。这 6 个字节中的前 4 个是 IP 地址，后两个是端口号。IP 地址和端口号均以大端的网络字节序表示。

注意： 网络字节序，简单地说就是网络上通过 socket 传输的字节序列。而字节序列就是字节顺序，它是指占内存多于一个字节类型的数据在内存中的存放顺序，通常有小端、大端两种字节顺序。小端字节序指低字节数据存放在内存低地址处，高字节数据存放在内存高地址处；大端字节序是高字节数据存放在低地址处，低字节数据存放在高地址处。

综上所述，默认情况下 peers 列表的长度是 50，如果 torrent 中的 peers 比较少，那么这个列表长度会更短；否则如果 torrent 中的 peers 比较多，那么 tracker 就得从这些 peers 中随机选择一些放入响应中。当响应请求时，Tracker 需要使用一个智能的机制来进行 peer

选择。例如，无须向做种者报告种子文件信息（reporting seeds to other seeders could be avoided）。

🔔注意：这里说的意思是，如果没有好的、较智能的机制来对 Peer 进行选择，就有可能出现 Tracker 向做种子的 Peer 报告种子信息，这很显然是不必要的。

如果有一个事件发生（即 stopped 或者 completed）或者客户端需要获得更多的 peers 时，客户端发送请求的频率肯定高于指定的间隔。尽管如此，不断地向 Tracker 发送请求以获得更多的 peers 绝对不是一个好主意。如果一个客户端想在其响应中包含一个更长的 peer 列表，那么它应该通过指定 numwant 参数来达到目的。

下面这段话是 BT 协议规范中对 BT 系统的开发实现进行的指点性说明。如果有读者从事 BT 开发，应该谨记这几句话。

30 个 peers 已经是一个很大的数目了，官方的客户端版本 3 在 peers 少于 30 的情况，会积极地向 Tracker 建立连接已获得更多的 peers，但是如果客户端已经有 55 个 peers 的情况下，客户端将拒绝和 Tracker 建立连接。这个值的选择对于性能非常重要。当一个片已经下载完成后，客户端需要将 HAVE 消息（参考下文 Peer 端协议的讲解）发送给大多数活跃的 peers。结果是广播通信的代价与 peers 数量成正比。这个数高于 25 之后，新加入的 peers 不太可能提升下载速度。强烈建议 UI 设计者使这个数变得模糊并且使之难以修改，因为这样做没有任何作用。

4．Tracker服务器的scrape的约定（Tracker 'scrape' Convention）

在 BT 中，大部分 Tracker 支持另一种形式的请求，这种方式查询 Tracker 所有给定的.torrent，即 Tracker 正在管理的所有.torrent。这种方式通常被称为 scrape page。

类似于上面描述的 URL，scrape URL 也是一个 HTTP GET 方法。尽管如此，它们的 Base URL 是不相同的。想要得到 scrape URL，就得使用如下步骤。

（1）从 announce URL 开始，找到该 announce URL 中的最后一个“/”。

（2）如果该“/”之后的内容不是 announce，那么说明这个 Tracker 不支持 scrape。

（3）如果这个 Tracker 支持 scrape，那么使用 scrape 代替 announce 就可以得到 scrape 页。

从 announce URL 到 scrape URL 的转换（announce URL -> scrape URL）

```
http: //example.com/announce          ->http: //example.com/scrape
http: //example.com/x/announce        ->http: //example.com/x/scrape
http: //example.com/announce.php      ->http: //example。com/scrape.php
http: //example.com/a                 ->不支持 scrape (scrape not supported)
http : //example.com/announce?x2%0644       ->http : //example.com/scrape?
x2%0644
http: //example.com/announce?x=2/4  ->不支持 scrape (scrape not supported)
http: //example.com/x%064announce     ->不支持 scrape (scrape not supported)
```

scrape URL 可以作为可选参数 info_hash（上文描述的一个 20 字节值）的一个补充。但是这样限制了 Tracker 向那个特殊的.torrent 发送报告；否则 tracker 正在管理的所有.torrent 统计数据将会返回。在实际开发 BT 系统的过程中，尽可能使用 info_hash 参数，这样就能

够减少 Tracker 的负载和带宽。

当然也可以指定多个 info_hash 参数给 Tracker （得支持多个 info_hash 参数）。这不是官方规范的一部分，但是已经成为了实际标准，例如这样一个 URL，http: //example.com/scrape.php?info_hash=aaaaaaaaaaaaaaaaaaaa&info_hash=bbbbbbbbbbbbbbbbbbbb&info_hash=cccccccccccccccccccc。

这个 HTTP GET 方法的响应是一个 text/plain 或者有时候是用 gzip 压缩的文本，这个文本由一个 B 编码的 dictionary 组成，这个 dictionary 包含如下键（key）。

files：这是一个 B 编码的 dictionary，在该 dictionary 中，每一个.torrent 文件都有其相应的键/值，这个键/值是相应.torrent 文件的统计数据。每一个键（key）由 20 字节的二进制 info_hash 组成。而该键所对应的值也是 dictionary，它包含如下的键（key）。

- complete：完成文件下载的 peer 数，即做种者的数量。值为整数类型。
- downloaded：已向 tracker 注册的下载完成的总次数（event=complete，即一个客户端完成了下载）。值为整数类型数据。
- incomplete：非做种的 peers 数（还没有完成该文件下载的 peers 数），即“占他人便宜者”。值为整数类型数据。
- name：（可选的）torrent 的内部名，由.torrent 文件中 info 键所对应值中的 name 指定。

例如，下面是响应有 3 层的 dictionary 嵌套，其 B 编码形式为：

```
d5：filesd20：…………d8：completei5e10：downloadedi50e10：incompletei10eeee。
```

在这个 B 编码的代码中省略号“…………”表示的是一个 20 字节的 info_hash。如果按 B 编码的规则将这些键与值解析出来就可以分析出，在这条响应消息中传递的信息是有 5 个做种者（complete，做种中），50 个已经完成下载的（downloaded，下载完成者）及 10 个正在下载者（incomplete，正在下载者）。这样，一个完整的响应信息就得到了。

5．scrape的非正式扩展（Unofficial extensions to scrape）

下面的响应键是非官方的。因为它们都是非官方的，因此都是可选的。

（1）failure reason：可读的错误信息，这个信息告诉客户端请求失败的原因（字符串类型）。使用该键的知名客户端有 Azureus。

注意：Azureus 是一款 BT 客户端软件，是由 Java 语言开发的免费、开源、功能强大的 BT 客户端程序，由开源社区负责维护和发布。

（2）flags：这是一个 B 编码的 dictionary，它包含多个标志。flags 键对应的值是另一个嵌套的 dictionary，可能包含如下键（key）。

- min_request_interval：这个键所对应的值是一个整数，该整数指定两个 scraping 操作之间的最小间隔秒数。发送该键（key）的知名 Tracker 是 BNBT。使用该键（key）的知名客户端是 Azureus。

6.3.7　BT 协议分析之——Peer 端协议（Peer wire protocol）

在 BT 协议规范中，Peer wire protocol 可理解为 Peer 端协议，也就是 Peer 之间进行交

互所使用并遵循的协议。BitTorrent 的 Peer 端协议，就是我们熟知的 TCP 协议。

1．关于Peer write protocol协议描述

peer（端）协议使片（piece）的交换变得容易，关于片的说明，请参见本节元信息文件的说明。

注意：原来的规范在描述 peer 协议时，也使用术语 piece（片），但是这不同于元信息文件里面的术语 piece（片），由于这个原因，在本规范中，将使用术语块（block）来描述 peers（端）之间交换的数据。

一个客户端（client）必须维持其与每一个远程 peer（端）连接的状态信息，这些信息由如下关键字描述。

（1）choked：远程 peer（端）是否已经开放了本客户端的访问权限。如果远程 peer（端）对本客户端是阻塞的，远程 peer 将不会接收来自本客户端的请求，而本客户端在接收到来自远程 peer 的阻塞（choke）消息后，就不会再试图向远程 peer 发送数据请求，因为本客户端会认为所有发给远程 peer 的请求已经被远程 peer 丢弃了。

（2）interested：远程 peer（端）是否对开放了本客户端提供的数据感兴趣。这是远程 peer 在通知本客户端，当本客户端不阻塞它们时，远程客户端将开始请求块（block）。

这也意味着本客户端需要记录它是否对远程 peer（端）感兴趣，以及它是否 choke/unchoke 远程 peer。因此真正的列表是如下这种表示方式。

- am_choking：本客户端正在 choke 远程 peer。
- am_interested：本客户端对远程 peer 感兴趣。
- peer_choking：远程 peer 正在 choke 本客户端。
- peer_interested：远程 peer 对本客户端感兴趣。

客户端连接开始时状态是 choke 和 not interested（不感兴趣），换句话就是：

- am_choking =1；
- am_interested=0；
- peer_choking=1；
- peer_interested=0。

当一个客户端对一个远程 peer 感兴趣并且该远程 peer 没有 choke 这个客户端，那么这个客户端就可以从远程 peer 下载块（block）。当一个客户端没有 choke 一个 peer，并且那个 peer 对这个客户端感兴趣时，这个客户端就会上传块（block）。

客户端必须不断通知它的 peers，是否对它们感兴趣，这一点是很重要的。客户端和每个端的状态信息必须保持最新，即使本客户端被 choke。这个允许所有的 peer 知道，当它们 unchoke 该客户端后，该客户端是否开始下载（反之亦然）。

2．数据类型

如果没有用其他的方法指定，在 peer wire 协议中的所有整数都会编码为 4 个字节的大端（big-endian）值。这也包括在握手之后，所有报文（Message）的长度前缀。

3．报文流（Message flow）

Peer wire 协议由一个初始的握手组成。握手之后，peers 通过以长度为前缀消息的交换

进行通信。长度前缀就是上面描述的整数。

🔔注意：因为 ICMP-Internet 控制报文协议中的 Message 翻译成报文，同时 IP/TCP 层中传输的数据都翻译为数据报，应用层传输的数据都翻译成报文，因此在这里 Message 翻译成报文，当然也可以理解为消息。

4．握手（HandShake）

握手是一个必需的报文，并且必须是客户端发送的第一个报文。该握手报文的长度是（49+len（pstr））字节。

握手的消息格式是 handshake：<pstrlen><pstr><reserved><info_hash><peer_id>。

- pstrlen：<pstr>的字符串长度，单个字节。
- pstr：协议的标识符，字符串类型。
- reserved：8 个保留字节。当前的所有实现都使用全 0。这些字节里面的每一个字节都可以用来改变协议的行为。来自 Bram 的邮件建议应该首先使用后面的位，以便可以使用前面的位来改变后面位的意义。
- info_hash：元信息文件中 info 键（key）对应值的 20 字节 SHA1 散列值。这个 info_hash 和在 tracker 请求中的 info_hash 是同一个。
- peer_id：用于唯一标识客户端的 20 字节字符串。这个 peer_id 通常与在 tracker 请求中传送的 peer_id 相同（但也不尽然，例如在 Azureus，就有一个匿名选项）。

🔔注意：在 BT 协议 1.0 版本中，pstrlen=19，pstr="BitTorrent protocol"。

连接的发起者应该立即发送握手报文。如果接收方能够同时地服务多个 torrent，它会等待发起者的握手报文（torrent 由 infohash 唯一标识）。尽管如此，一旦接收方看到握手报文中的 info_hash 部分，接收方必须尽快响应。tracker 的 NAT-checking 特性不会发送握手报文的 peer_id 字段。

如果一个客户端接收到一个握手报文，并且该客户端没有服务这个报文的 info_hash，那么该客户端必须丢弃该连接。

如果一个连接发起者接收到一个握手报文，并且该报文中 peer_id 与期望的 peer_id 不匹配，那么连接发起者应该丢弃该连接。注意发起者可能接收来自 tracker 的 peer 信息，该信息包含 peer 注册的 peer_id。来自于 tracker 的 peer_id 需要匹配握手报文中的 peer_id。

5．关于peer_id

peer_id 长 20 个字节。至于怎么将客户端和客户端版本信息编码成 peer_id，现在主要有两种惯例：Azureus 风格和 Shadow 风格。

（1）Azureus 风格使用如下编码方式：‘-’，紧接着是 2 个字符的 client id，再接着是 4 个数字的版本号，‘-’，后面跟着随机数。

例如：“-AZ2060-”……

使用这种编码风格的还有很多其他的知名客户端，这里就不再一一列举了，详情请自行参考 BT 协议规范中的说明。

（2）Shadow 风格使用如下编码方式：一个用于客户端标识的 ASCII 字母数字，多达 5

个字符的版本号（如果少于 5 个，则以“-”填充），紧接着是 3 个字符（通常是“---”，但也不总是这样），最后跟着随机数。版本字符串中的每一个字符表示一个 0～63 的数字。

注意：Shadow 也是一款 BT 的客户端软件，它的详细表示方法请参考 BT 协议规范。

关于 Shadow 的这种编码风格，目前也有很多 BT 客户端在使用，常用的客户端有：

'A'–ABC，'O'-Osprey Permaseed，'Q'–BTQueue，'R'–Tribler，'S'-Shadow's client，'T'–BitTornado，'U'-UPnP NAT Bit Torrent 等。

- BitComet 使用不同的编码风格。它的 peer_id 由 4 个 ASCII 字符 exbc 组成，接着是 2 个字节的 x 和 y，最后是随机字符。版本号中的 x 在小数点前面，y 是版本号后的两个数字。BitLord 使用相同的方案，但是在版本号后面添加 LORD。BitComet 的一个非正式补丁曾经使用 FUTB 代替 exbc。自版本 0.59 开始，BitComet peer id 的编码使用 Azureus 风格。
- XBT 客户端也使用其特有的风格。它的 peer_id 由 3 个大写字母 XBT 以及紧随其后代表版本号的 3 个 ASCII 数字组成。如果客户端是 debug 版本，第七个字节是小写字符 d，否则就是“-”。接着就是“-”，然后是随机数，大写和小写字母。例如：peer_id 的开始部分为“XBT054d-”表明该客户端是版本号为 0.5.4 的 debug 版本。
- Opera 8 预览版和 Opera 9.x 发行版使用以下的 peer_id 方案：开始的两个字符是 OP，后面的 4 个数字是开发代号。接着的字符是随机的小写十六进制数字。
- MLdonkey 使用如下的 peer_id 方案：开始的字符是“-ML”，后面跟着点式版本，然后就是一个“-”，最后跟着随机字符串。例如“-ML2.7.2-kgjjfkd”。
- Bit on Wheels 使用模式“-BOWxxx-yyyyyyyyyyyy”，其中 y 是随机的（大写字母），x 依赖于版本。如果版本为 1.0.6，那么 xxx = AOC。
- Queen Bee 使用 Bram 的新风格：‘Q1-0-0---’或者‘Q1-10-0-’之后紧随着随机字节。
- BitTyrant 是 Azureus 的一个分支，在它的 1.1 版本中，其 peer id 使用 AZ2500BT＋随机字节的方式。
- TorrenTopia 版本 1.90 自称是或源自于 Mainline 3.4.6，它的 peer ID 以“346------”开始。
- BitSpirit 有几种编码 peer ID 的方式。一种模式是读取它的 peer ID，然后使用开始的 8 个字节作为它的 peer ID 基础来重新连接。它的实际 ID 使用'\0\3BS'（c 标记法）作为版本 3。x 的前 4 个字节，使用'\0\2BS'作为版本 2.x 的前 4 个字节。所有方式都使用 UDP0 作为结尾。
- Rufus 使用它的十进制 ASCII 版本值作为开始的两个字节。第 3 个和第 4 个字节是'RS'。紧随其后的是用户的昵称和一些随机字节。
- C3 Torrent 的 peer ID 以-G3 开始，然后追加多达 9 个表示用户昵称的字符。
- FlashGet 使用 Azureus 风格，但是前面字符是 FG，没有“-”。版本 1.82.1002 仍然使用版本数字 '0180'。
- BT Next Evolution 源自于 BitTornado，但是试着模仿 Azureus 风格。结果是它的 peer ID 以-NE 开始，接着是 4 个数字的版本号，最后就是以 shadow peer id 风格描述客

户端类型的 3 个字符。

- Qvod 的 ID 以 4 个字母 QVOD 开始，接着是 4 个十进制数字的开发代号（目前是“0054”）。最后的 12 个字符是随机的大写十六进制数字。中国有一个修改版，该版本以随机字节代替前 4 个字符。

许多客户端全部使用随机数或者随机数后面跟 12 个全 0（像 Bram 客户端的老版本）。

以上讲述了不同的 BT 客户端其 Peer_id 的表示方式，在进行 BT 系统开发时，这是必须要注意的一点，详细的说明请参见 BT 协议规范原文。

6．报文（Messages）

BT 协议的所有报文采用如下的结构：<length prefix><message ID><payload>。length prefix（长度前缀）是一个 4 字节的大端（big-endian）值。message ID 是单个十进制值。playload 与消息相关。

- keep-alive：<len=0000>

keep-alive 消息是一个 0 字节的消息，将 length prefix 设置成 0。没有 message ID 和 payload。如果 peers 在一个固定时间段内没有收到任何报文（keep-alive 或其他任何报文），那么 peers 应该关掉这个连接。因此如果在一个给定的时间内没有发出任何命令的话，peers 必须发送一个 keep-alive 报文保持这个连接激活。通常情况下，这个时间是 2 分钟。

- choke：<len=0001><id=0>

choke 报文长度固定，并且没有 payload。

- unchoke：<len=0001><id=1>

unchoke 报文长度固定，并且没有 payload。

- interested：<len=0001><id=2>

interested 报文长度固定，并且没有 payload。

- not interested：<len=0001><id=3>

not interested 报文长度固定，并且没有 payload。

- have：<len=0005><id=4><piece index>

have 报文长度固定。payload 是 piece（片）的从零开始的索引，该片已经成功下载并且通过 hash 校验。

实际上，一些客户端必须严格实现该定义。因为 peers 不太可能下载他们已经拥有的 piece，一个 peer 不应该通知另一个 peer 它拥有一个 piece，如果另一个 peer 拥有这个 piece。最低限度“HAVE suppresion”会使用 have 报文数量减半，总地来说，大致减少 25%～35% 的 HAVE 报文。同时，给一个拥有 piece 的 peer 发送 HAVE 报文是值得的，因为这有助于决定哪个 piece 是稀缺的。

（1）bitfield 报文可能仅在握手序列发送之后，其他消息发送之前立即发送。它是可选的，如果一个客户端没有 piece，就不需要发送该报文。Bitfield 的报文格式是：

bitfield：<len=0001+X><id=5><bitfield>

bitfield 报文长度可变，其中 x 是 bitfield 的长度。payload 是一个 bitfield，该 bitfield 表示已经成功下载的 piece（片）。第一个字节的高位相当于 piece 索引 0。设置为 0 的位表示一个没有的 piece，设置为 1 的位表示有效的和可用的 piece。末尾的冗余位设置为 0。

长度不对的 bitfield 将被认为是一个错误。如果客户端接收到长度不对的 bitfield 或者

bitfield 有任一冗余位集，它应该丢弃这个连接。

（2）request 报文长度固定，用于请求一个块（block）。Request 的报文模式如下。

request：<len=0013><id=6><index><begin><length>，此报文的有效载荷（payload）信息包含如下信息。

- ❑ index：整数，指定从 0 开始的 piece 索引。
- ❑ begin：整数，指定 piece 中从 0 开始的字节偏移。
- ❑ length：整数，指定请求的长度。

（3）piece 报文长度可变，piece 报文格式如下。

piece：<len=0009+X><id=7><index><begin><block>，其中 x 是块的长度。此报文的 payload 包含如下信息：

- ❑ index：整数，指定从 0 开始的 piece 索引。
- ❑ begin：整数，指定 piece 中从 0 开始的字节偏移。
- ❑ block：数据块，它是由索引指定的 piece 的子集。

（4）cancel 报文长度固定，用于取消块请求。Cancel 的报文模式如下。

cancel：<len=0013><id<=8><index><begin><length>，此报文的 playload 与 request 报文的 playload 相同。一般情况下用于结束下载。

（5）port 报文由新版本的 Mainline 发送，port 的报文格式如下。

port：<len=0003><id=9><listen-port>，在新版本的 Mainline 中，实现了一个 DHT tracker。该监听端口是 peer 的 DHT 结点正在监听的端口。这个 peer 应该插入本地路由表（如果支持 DHT tracker 的话）。

BT 协议规范是整个 BT 技术的核心和基础，本节对 BT 协议规范的内容进行了重点分析，读者要重点理解元信息文件的结构及 Tracker 端和 Peer 端的协议。

注意：以上文章中关于 BT 协议规范的描述，均来自对原版英文 Bittorrent Protocol Specification v1.0 的翻译，个别地方做了标注和说明，如有任何疑问，请参考 BT 协议规范的原文。

6.4　如何使用 BT

在以上几节中，对 BT 的基础知识、下载技术及协议规范都进行了讲解和分析，从理论的层面上对 BT 技术体系有了整体的了解。下面就从应用的层面上来讲解一下如何使用 BT。任何一种技术都是为应用服务的，在理论知识的基础上再结合实际应用，能更好掌握 BT 技术。本节将重点讲解 BT 的应用。

6.4.1　BT 软件知多少

要想用好 BT，首先就要选择一个好的 BT 软件，这个软件就是 BT 下载客户端，目前互联网上有数十款功能各异的 BT 软件，这些软件大部分都是免费软件，有的还直接开放源代码，这里只简要列举并介绍几个目前较常用的 BT 客户端软件。

BitComet 比特彗星：BitComet 比特彗星是基于 BitTorrent 协议的 P2P 免费软件，在现在的 BT 客户端软件中占有较重要的地位，尤其是国内用户很多。BitComet 具有高效的网络内核、支持对一个 Torrent 中的文件有选择地下载、磁盘缓存技术、文件续传、支持多 Tracker 协议、多语言界面等多重功能。

µTorrent：最简约和有效率的 BT 客户端。µTorrent 很小巧，内容简约，设置简单，拥有 BT 客户端软件常见的功能，但是不支持 UDP 的连接协议。内网下载方面也有不错的表现，支持 IPV6，支持多任务同时下载，支持设置文件下载优先级，可以根据计划任务调整占用的带宽，全局/单个任务的速度限制，快速断点续传机制，支持 UPnP 端口映射，支持流行的 BT 扩展协议，支持用户来源交换，支持 DHT 等，最小内存占用仅 6MB。在互联网中有很大的用户群，也是研究 BT 技术的首选。

BT Plus!：BT Plus!原名为 BitTorrent Plus!，它是一个功能完善的 BT 客户端。它基于成熟的 BT 核心并继承和增强了 BitTornaolo 一系列功能。BitTorrent Plus!可以很容易地被自定义，有许多选项提供，目前已有约两千万的较大用户群。在 BT 下载中也是一款比较受欢迎的客户端软件。

下面还有一些 BT 客户端软件，这里只简单地将它们列举出来，就不再一一介绍了。读者可以根据这些软件的名字，自行参考相关资料了解它们的性能和特点。

这些软件有 BitLord、BitTornado、Vuze（原名 Azureus）、Transmission、比特精灵，Flashget、rTorrent、迅雷、超级旋风、Mldonkey、Deluge、KTorrent、FrostWire 等。

以上这些 BT 软件，其下载的原理和实现的思想基本都是一致的，只是在功能设置、界面外观和一些应用环境上有一定的区别，读者可以根据自己的应用特点和爱好选择自己喜欢的 BT 软件即可。

在当前国内的互联网用户中，BT 软件客户端使用最多的就是 BitComet 比特彗星，下面就以这款客户端软件为例，详细介绍在互联网条件下如何使用 BT、如何用好 BT。

6.4.2　BitComet 软件安装

BitComet 客户端软件是通过网络进行发布的，通过搜索引擎，很多站点都提供 BitComet 的免费下载链接。在网址为 http: //www.bitcomet.com/doc/download-achive-zh.htm 的网站中，可以找到所有版本的下载地址。在联网的条件下，直接单击下载地址就可以将 BitComet 软件包下载到本地。

BitComet 客户端软件包有两种安装模式，一种是安装模式，另一种是绿色免安装模式。安装版的软件包是 EXE 后缀，而免安装版的软件包是 ZIP 后缀的文件。

1．BitComet免安装版

下载好 ZIP 后缀的软件包后，直接将其解压到目标路径。在目标路径下，会生成两个文件，一个是 BitComet_Win9x.exe，另一个是 BitComet.exe。

（1）BitComet_Win9x.exe 是操作系统为 Win 98 \Win Me 的用户使用。

（2）BitComet.exe 是操作系统为 Win 2000 及以上的用户使用。

注意：免安装版的卸载，只需把 BitComet 文件所在目录删除就可以了。

2．BitComet安装版

（1）安装形式

- 没有安装 BitComet 任何版本的电脑或安装了 BitComet 后，卸载且删除残留文件的电脑，进行全新安装。
- 已经安装了 BitComet 的电脑，进行覆盖安装。

注意：两种安装形式没有分别，都属于正常安装。其中覆盖安装保留了任务列表和设置文件，免除了重新设置的麻烦。

（2）安装前的准备

若已经安装了旧版 BitComet，请先关闭 BitComet。本文演示的安装过程，是 BitComet1.14 版本的安装过程，此版本也是当前的最新版本，可能由于版本更新出现一些与本文描述的安装过程不一致的地方，读者需要注意新版本的更新说明。

注意：关闭 BitComet 时，会同时关闭 BitComet 资源浏览器，请注意妥善处理未完成的工作。

3．BitComet的安装过程

BitComet 的安装就如同普通的软件安装一样，下面简要说明一下 BitComet 1.14 版本的 BT 下载客户端在 Windows 系统上的安装过程。双击下载好的 BitComet.exe 安装软件包，开始进行安装。

（1）选择安装语言

安装过程中，首先会弹出对话框让用户选择安装语言，如图 6.16 所示。

注意：因为操作系统语言的不同，下拉列表框中有的选项为“乱码”，属于正常现象，不影响安装、使用。

设置完安装语言后，进入安装向导，如图 6.17 所示。



图 6.16　BitComet 安装过程中选择安装语言对话框

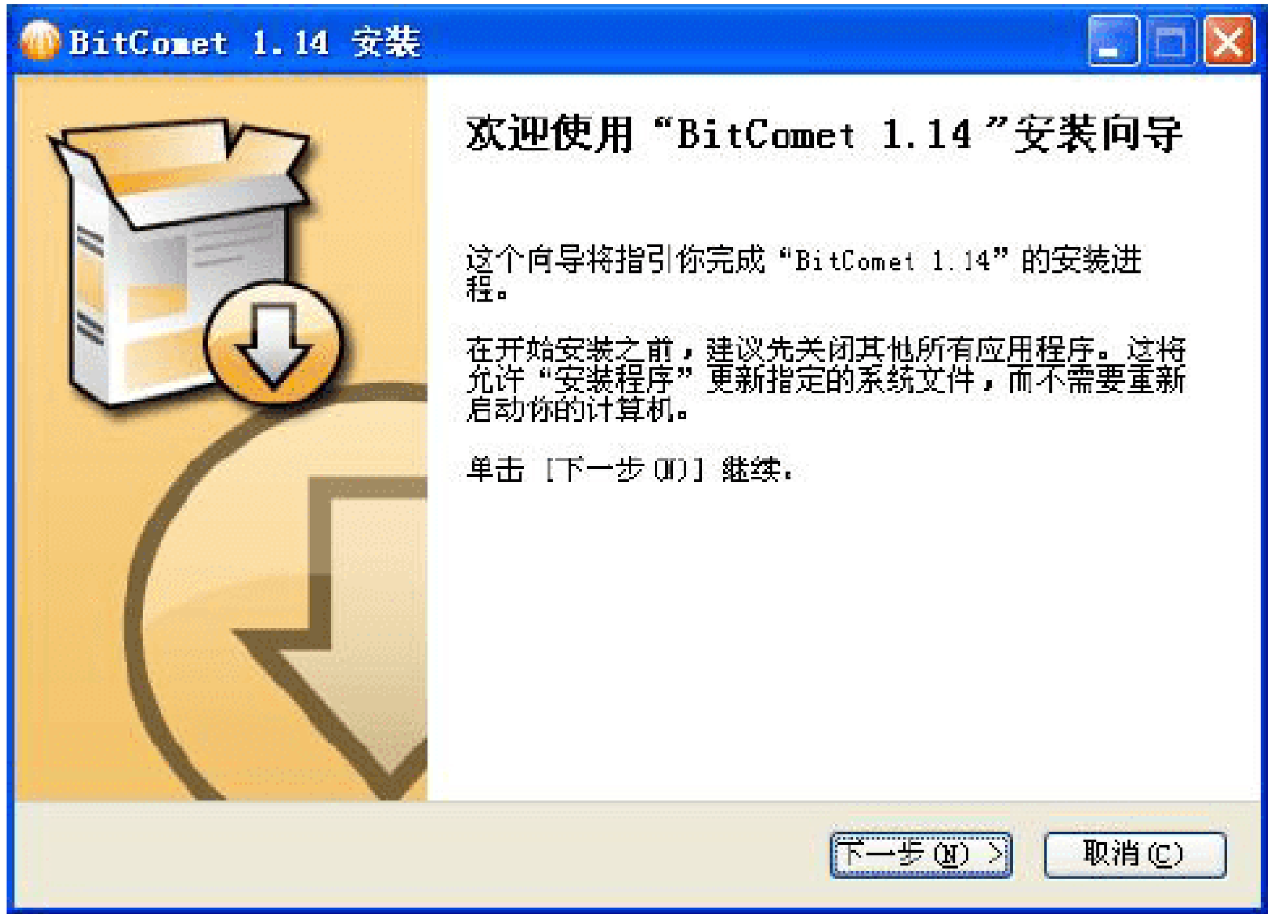


图 6.17　BitComet 1.14 安装向导对话框

图 6.17 所示的安装对话框中，单击“下一步”按钮，弹出现软件安装许可协议和隐私

政策说明对话框。在这两个过程中，只需单击“我接受”和“下一步”按钮，就会出现如图 6.18 所示的“选择组件”对话框。

（2）选择所需组件

选择组件，主要有以下几项。

- 开始菜单项：是否在开始菜单中加入 BitComet 项。
- 桌面快捷方式-BitComet 主程序：BitComet 主程序是否创建桌面快捷方式。
- 桌面快捷-BC 资源浏览器：BitComet 资源浏览器是否创建桌面快捷方式。
- 浏览器集成包含：捕获 IE 文件下载：在 IE 中单击下载链接，会直接用 BitComet 下载；添加到 Firefox 右键菜单：在 Firefox 右键菜单中添加 BitComet 项；遨游（Maxthon）浏览器集成：在 Maxthon 右键菜单中添加 BitComet 项，并集成捕获功能。

注意：若没有安装 Firefox、Maxthon，则相应的组件会灰化，即处于不可选状态。

对于这些组件的设置，基本上都选择默认的设置，如有特别的需要，只需根据相关选项进行设定即可。

（3）选择安装目录

对相关组件完成设置后，就会出现“选择安装目录”的界面，如图 6.19 所示。

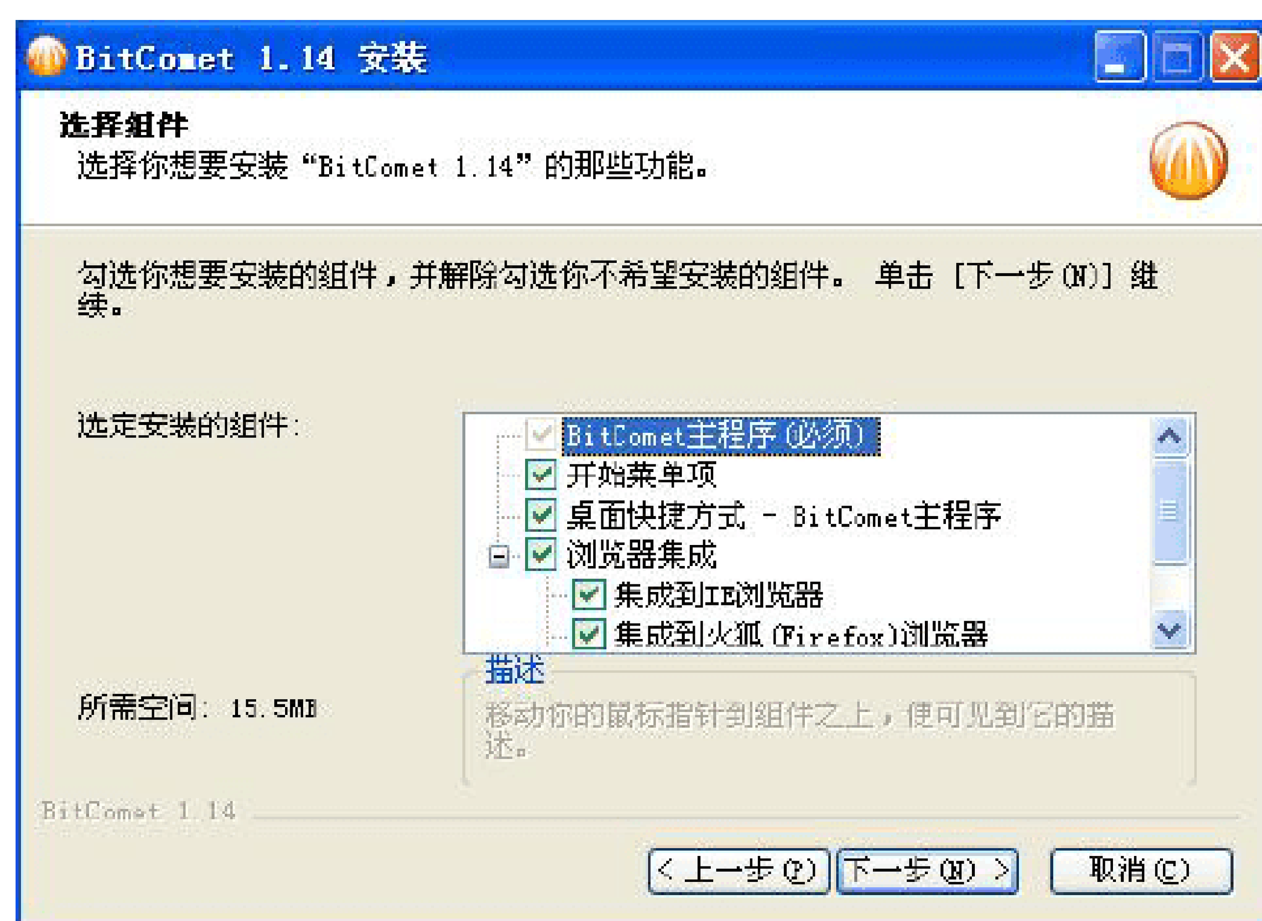


图 6.18　BitComet 安装中选择组件界面

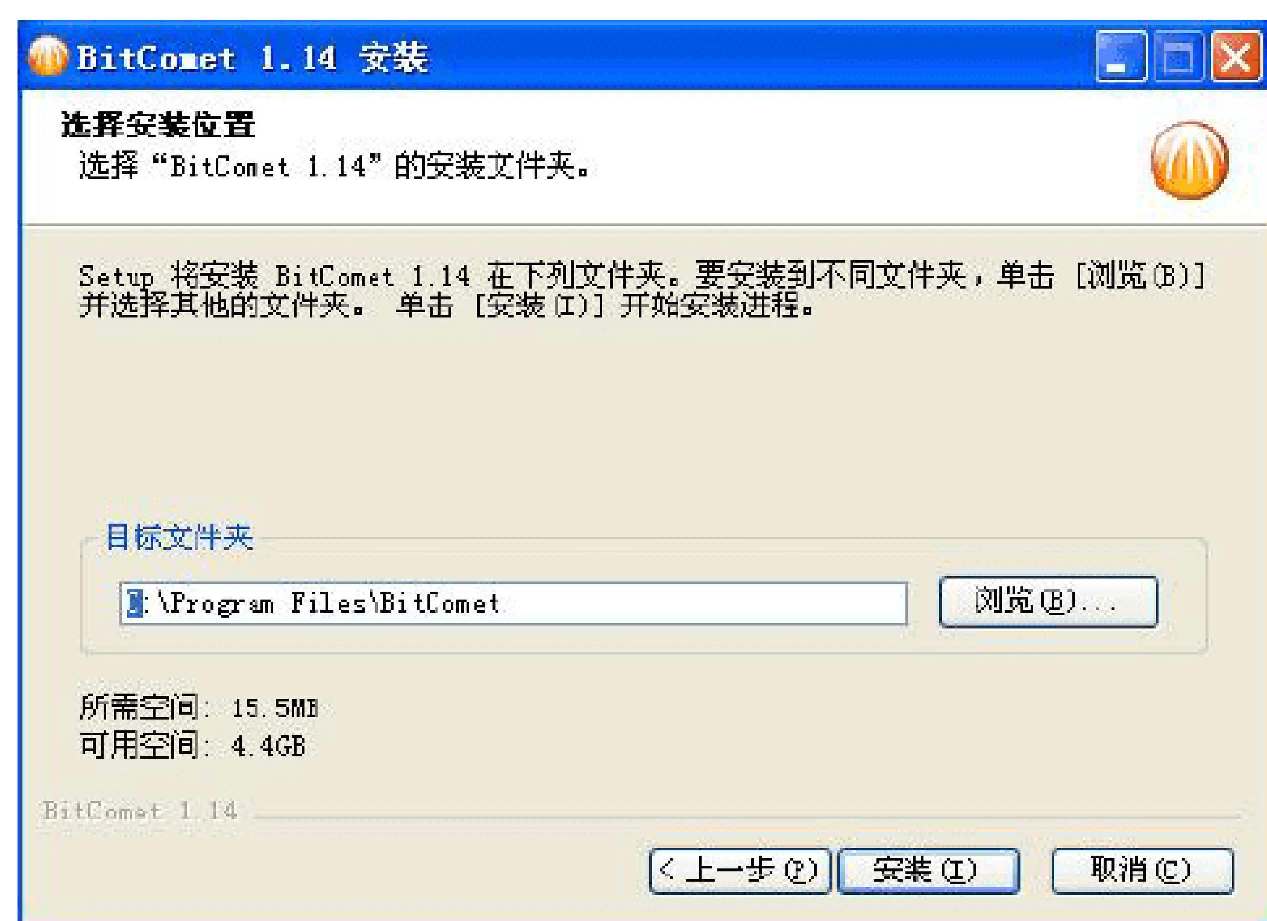


图 6.19　选择安装路径界面

在 6.19 所示的路径选择界面中，默认安装路径为 C:\Program Files\BitComet\，推荐读者在安装时，尽量将其安装在非系统盘下。

注意：Bitcomet 为绿色软件，系统重装后仍可继续使用，覆盖安装也不会清除以前的设置。

（4）释放文件执行安装

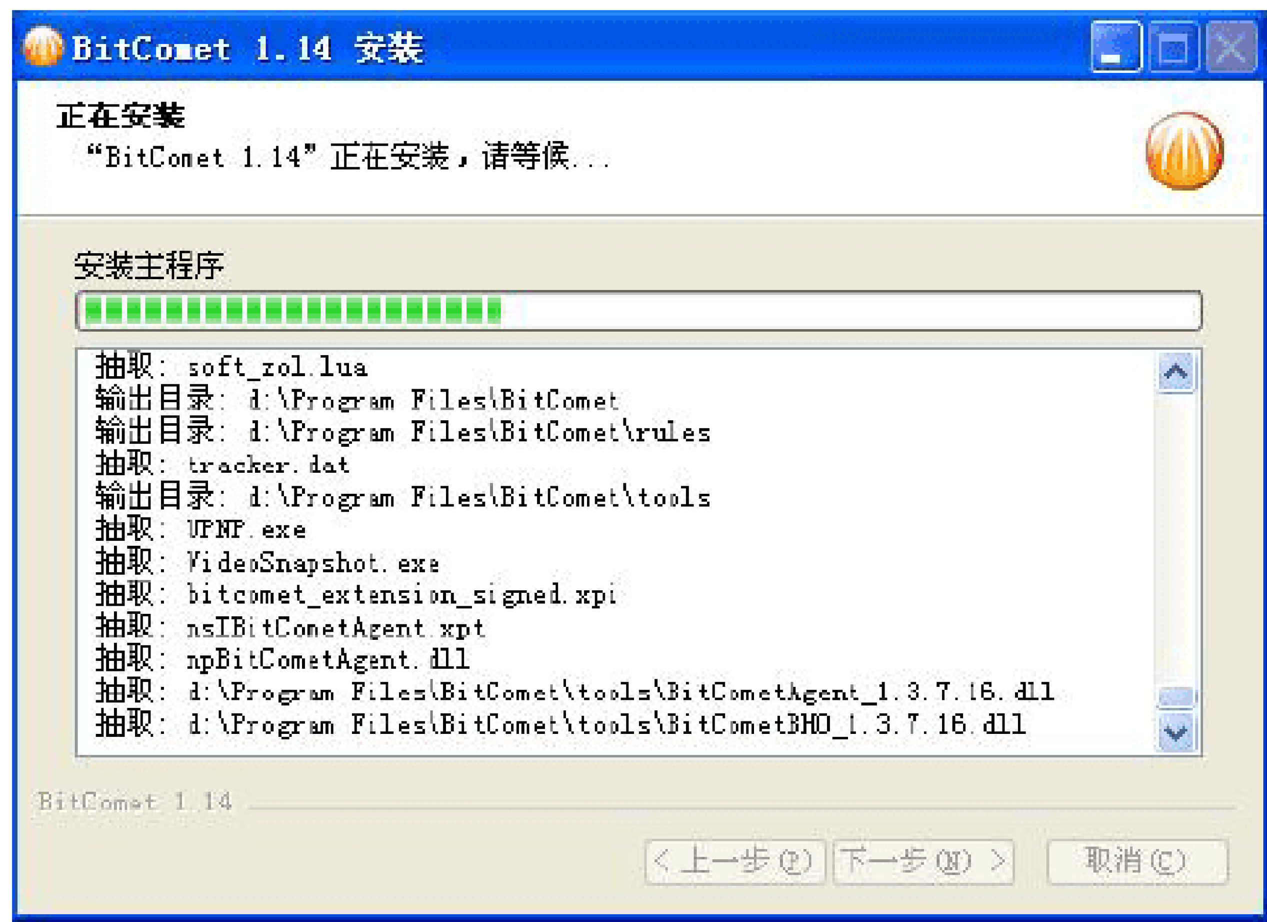
选择完安装路径，就执行文件的释放过程，如图 6.20 所示。

在文件释放的过程中，如果覆盖安装，且安装前没有关闭正在运行的 BitComet，那么安装过程会自动将其关闭。但在文件释放过程可能会出现如图 6.21 所示的错误提示，直接“重试”几次，就可以了。

（5）安装结束

在安装过程中，一直单击“下一步”按钮，直到安装完成。安装成功后，会出现如图

6.22 所示的界面。



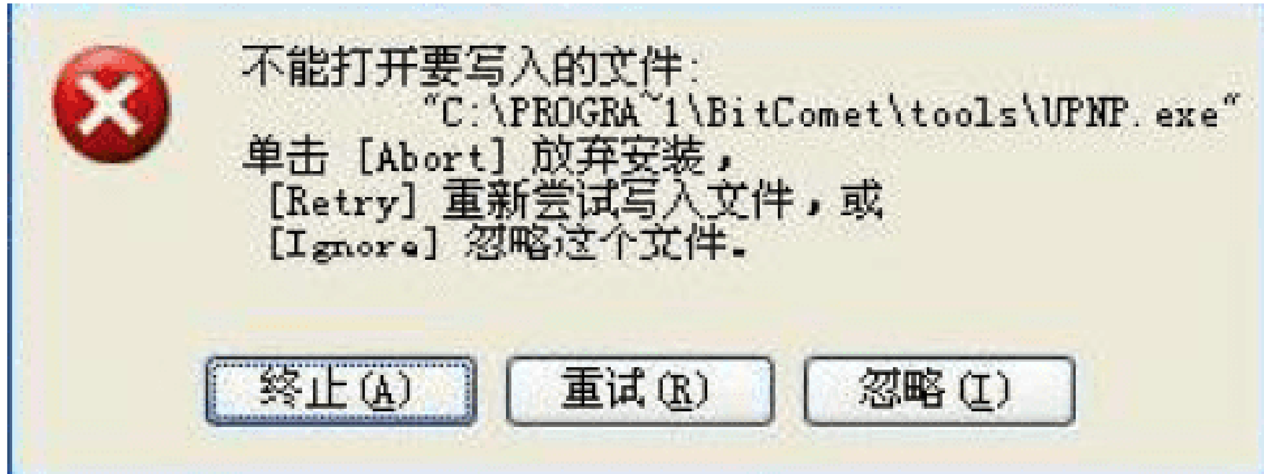


图 6.20　BitComet 安装中文件的释放过程　　　　图 6.21　安装错误提示界面



图 6.22　BitComet 安装完成的提示界面

在图 6.22 所示的界面中有 3 个选项，分别如下。

- 运行 BitComet 1.14（B）：单击"完成"按钮后就运行 BitComet 1.14 客户端程序。
- 开机后自动运行 BitComet：就是在每次开机时，会自动运行 BitComet，此设置也可以在 BitComet 选项中进行重置。
- 显示"自述文件"：打开 BitComet 1.14 的描述信息，如更新信息、功能描述等。

在图 6.22 所示的界面中，单击"完成"按钮，BitComet 1.14 的整个安装过程就全部完成了。

6.4.3　使用 BitComet 进行 BT 下载

使用 BitComet 可以进行多种下载，下载 BT 是其最核心的应用，当然也可以下载 HTTP、FTP 等。本节主要讲 BT 的下载方式，在 BitComet 中，先获取.torrent 的种子文件，然后打开这个种子文件就可以进行 BT 下载了。

1．获取.torrent种子文件

通过.torrent 文件进行 BT 下载，是最常用的下载方式，基本过程就是先下载一个.torrent

文件，然后在 BitComet 中打开这个文件就可以进行 BT 下载了。通过搜索引擎，在当前的网络条件下，可以很容易地搜索到想要下载的 BT 种子文件。如果不知道从哪里获得.torrent 文件，可以参考图 6.23，有两种方法可以获得.torrent 文件。

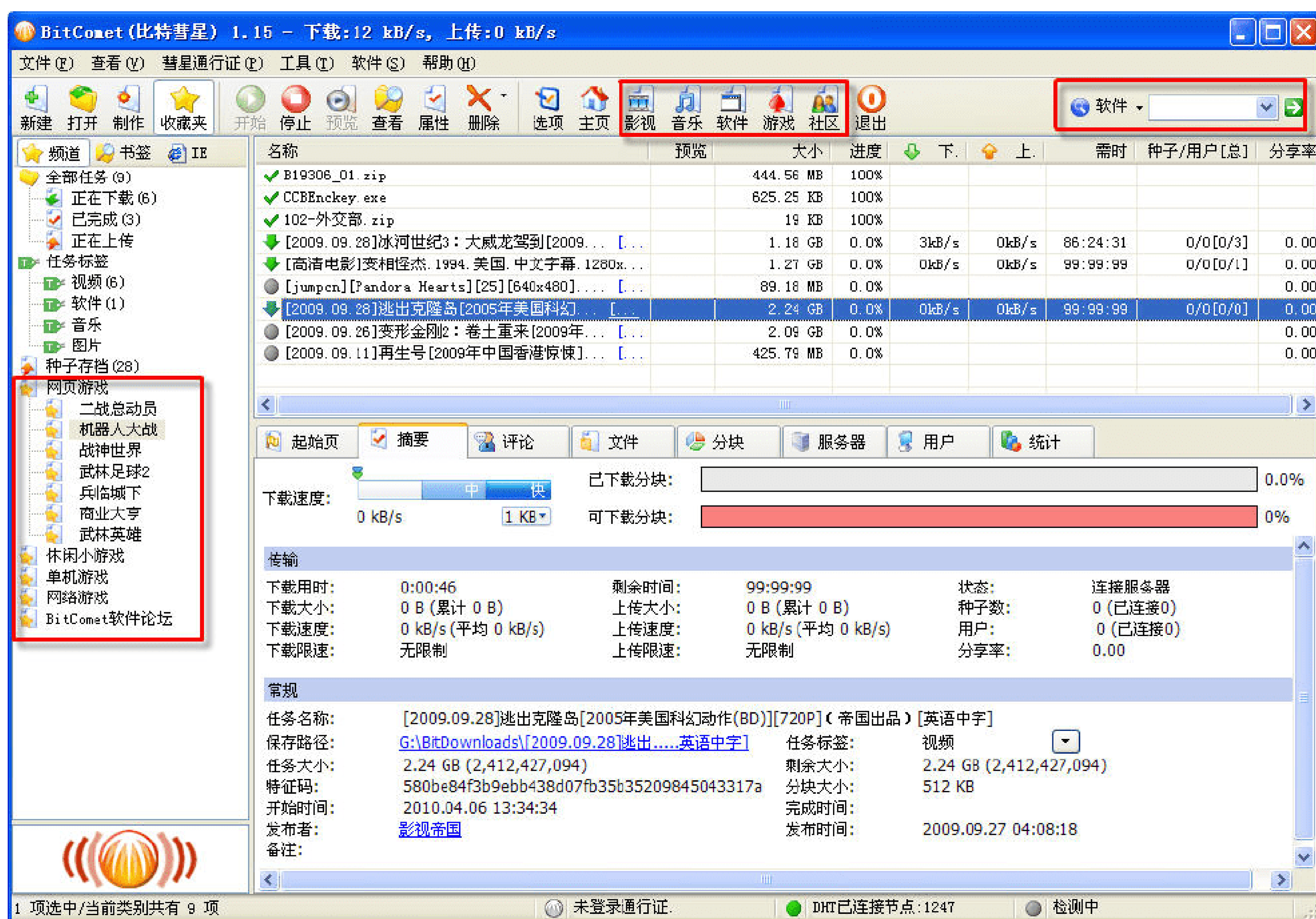


图 6.23　获取.torrent 文件的方法

在图 6.23 所示的方法中，方法一是先打开 BitComet 软件，选择界面左侧的“频道”标签，就会出现一个列表。在此列表中有一个选项叫“BT 发布站点”，单击展示此选项的列表就可以看到很多 BT 网站。可以在这些 BT 站点上下载.torrent 文件。方法二是先打开 BitComet 软件，在工具栏搜索条中，直接填入关键字，单击“搜索”按钮，在电影地带页面中会出现搜索结果的.torrent 文件列表。

2．向BitComet中加入BT任务

向 BitComet 中加入 BT 任务进行下载，可以有 3 种方式。

- 打开 BitComet 主界面，在系统工具菜单中选择“文件”|“打开 Torrent 文件”或 Ctrl+O 快捷键。通过浏览的方式，打开已经下载到本地的 Torrent 文件，就会弹出 BT 任务下载对话框，在其中选择保存路径、下载文件就可以下载了。
- 双击下载到本地的 Torrent 文件，系统会直接打开 BitComet 程序，然后弹出 BT 任务下载对话框，再进行 BT 任务下载。
- 将 Torrent 文件拉到 BitComet 主程序中或悬浮窗口中，弹出 BT 任务下载对话框，再进行 BT 任务下载。

3．BT任务下载对话框参数设置

打开 Torrent 文件时，会弹出 BT 任务下载对话框，其中会有一些基本选项需要设置，设置界面如图 6.24 所示。

主要的设置选项如下。

- ❑ 保存位置：单击浏览器或者直接在地址栏中输入地址，可以改变文件下载后存储在电脑里的位置。
- ❑ “...”：3 个点的省略号，为选择保存路径的操作按钮。
- ❑ 预设：设置保存路径的选择规则。使用默认保存目录或按类别选择目录。用户可根据需要自行设定。
- ❑ 任务类别：可以选择已经存在的类别，也可以新建类别。
- ❑ 立即开始/手动开始：意思是立刻开始下载这个任务。如果选择手动开始，则需要手动开始任务下载。
- ❑ 文件名：通过在文件名前面打勾，可以选择性地下载文件。
- ❑ 缺省：设置当前的保存路径、保存路径选择机制、启动机制为默认值。
- ❑ Torrent 内容：包含名称、发布者、文件大小、磁盘空间及下载文件。下载文件可以选择，在想要下载的文件前勾选即可。

注意： 一般只需要改变保存位置和需要下载的文件，其余选项可以不作改变，保持默认值（这些值都是开发人员帮您选定的最优设置）。直接单击“确定”按钮就可以开始任务下载了，就可以关闭当前阅读的页面，享受 BitComet 带来的快速下载体验。当然如果有兴趣，可以继续往下阅读，了解高级应用。

在图 6.24 所示的界面中，还有“高级设置”选项卡，打开后，如图 6.25 所示。



图 6.24　BT 任务下载设置界面

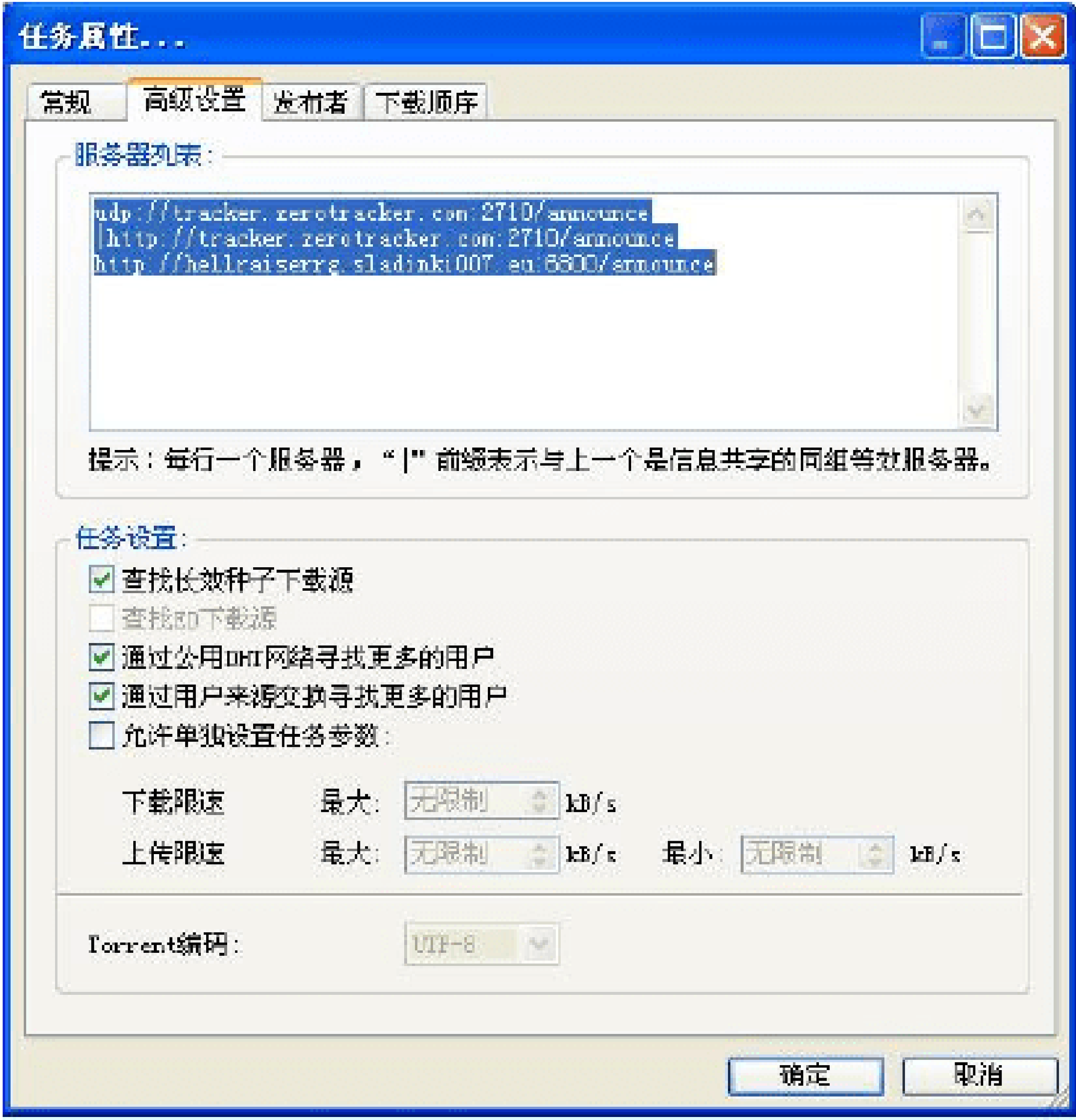


图 6.25　BT 下载任务中高级设置界面

在这个高级选项设置中，主要有以下设置项。

- ❑ 服务器列表：里面填入的是 tracker 服务器地址，这些地址一般是.torrent 文件里自带的，也是做种子的时候填入的，下载的时候不用关注。

- 任务设置：查找长效种子下载源，查找可以长时间做种的下载源。

注意：长效种子，一方面从完成度为 100%用户那里获得数据，另一方面在网络上寻找镜像服务器并获得数据，这样能大大增加下载速度。但在增加下载速度的同时，也要上传数据给其他用户。对于发布时间间隔比较久的 BT 任务，很多时候都会遇到没有种子的情况。没有种子，自然就不能完成任务。长效种子功能能够让完成度为 100%用户后台做种，省去了补种的麻烦。

- 查询 ED 下载源：可以从 ED 插件处获得加速，需要安装 ED 插件，否则此项灰化。
- 允许使用公用 DHT 网络寻找更多用户：可以使 DHT 网络上的其他用户获得数据，提高下载速度。
- 允许用户来源交换寻找更多用户：可以帮助获得更多的用户从而获得更多的数据，提高下载速度。
- 允许单独设置任务参数：可以设置最大下载速度、最小保证上传速度和最大允许上传速度。一般下载用户主要是为了做种子用户服务的，作为种子，用户通过设置这个值即可控制自己上传数据的最小值和最大值，从而可以保证其他用户下载的速度，并且预留出部分带宽供自己下载文件。

注意：高级设置中的数据一般不用修改就可以保证最好的下载设置。

在图 6.25 所示的界面中，还有发布者选项卡，在这个选项卡中，主要是记录发布者的相关信息，以感谢每一位种子发布者。最后一个，是下载顺序选项卡，打开这个选项卡后，如图 6.26 所示。

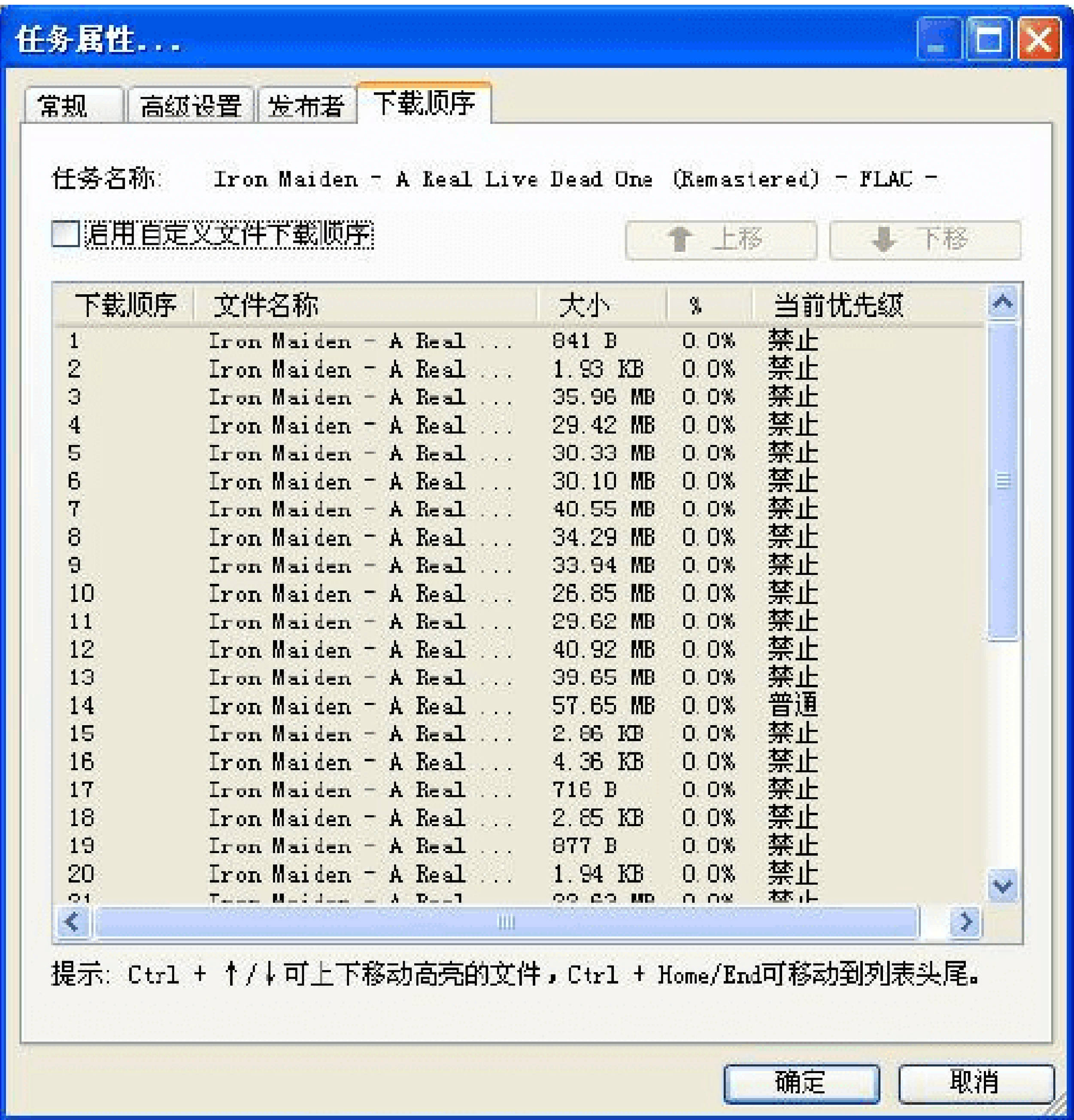


图 6.26　BT 下载中下载顺序设置界面

在图 6.26 所示的界面中，有一个复选框，“启用自定义文件下载顺序”，选中启用此功能后，就可以按照列表中的顺序设置文件下载的优先级别，从而达到顺序下载的目的。

按照以上的过程，并进行相关选项的设置，就可以用 BitComet 进行 BT 下载了。

6.4.4　利用 BitComet 制作 Torrent 文件

制作 Torrent 文件过程，就是将文件的信息进行 B 编码的过程，BitComet 提供了这种编码机制，可以简单地将一个或多个资源文件制作成 torrent 文件。有了这个 Torrent 文件，就可以在 Web 网络上将其发布出去，一旦有人下载这个文件，那么这个 Torrent 所表征的文件资源也就共享到 BT 网络中去了。

1．Torrent文件制作方法

利用 BitComet 软件，可以将各种资源文件制作成.torrent 文件格式的元文件以进行发布。下面就讲一下在 BitComet 中制作 Torrent 文件的方法。

打开 BitComet 的主界面，在主界面的菜单栏中，选择“文件”|“制作 Torrent 文件”命令或者单击图标菜单项中的“制作”按钮，打开制作 Torrent 文件窗口，如图 6.27 所示。

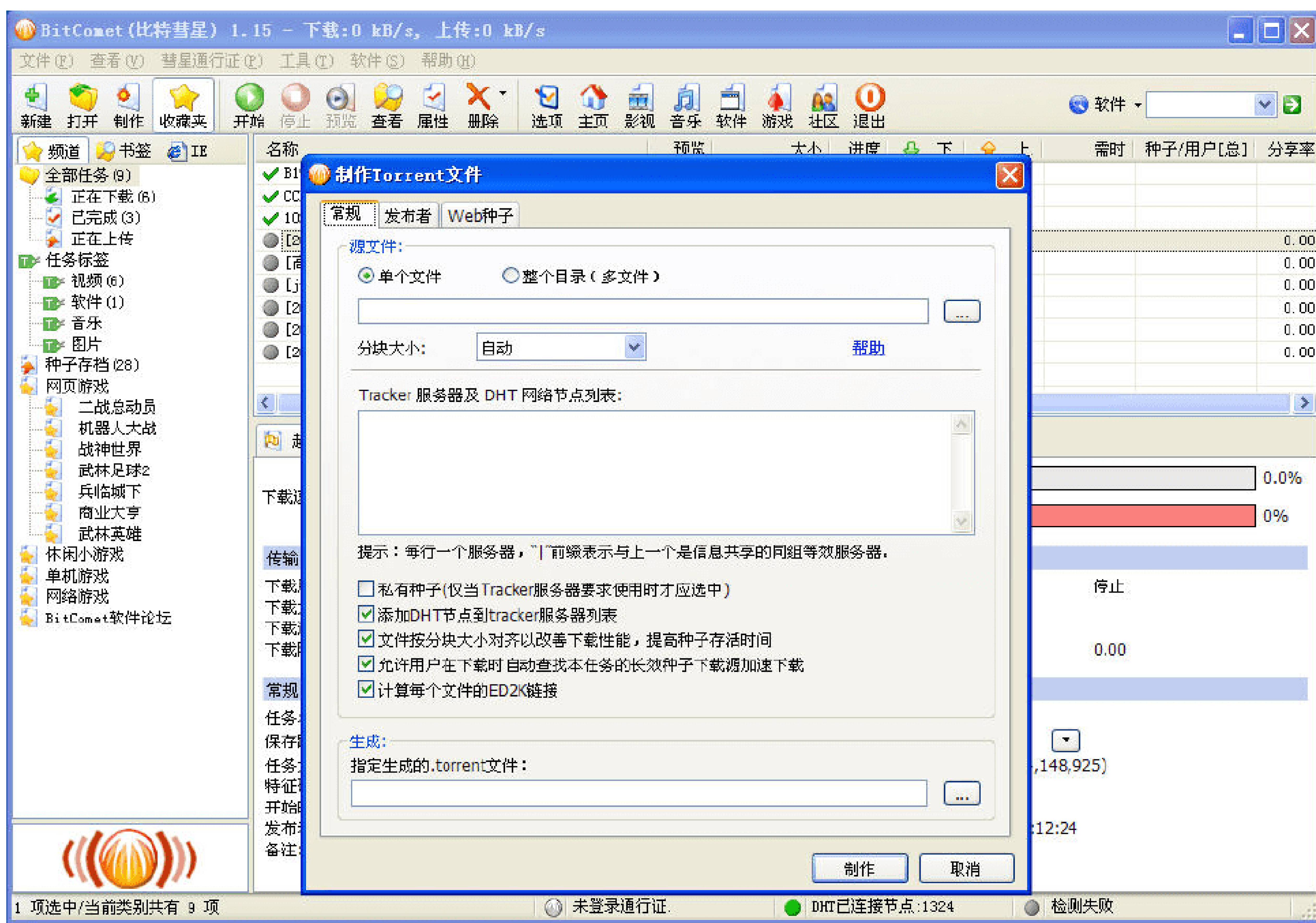


图 6.27　在 BT 中制作 Torrent 文件界面

在图 6.27 所示的“制作 Torrent 文件”选项卡中，主要有以下几个选项。

- 源文件：选择要将“单个文件”还是“整个目录（多文件）”制作成.torrent 文件。然后单击浏览选择文件或者目录。
- 分块大小：BT 发布的时候，都是将发布的资源分成一个个的块（piece）。这里的分块大小，指的是将所有发布的资源分成块后每一块的大小。只需选择默认设置即可，系统自动选择分块大小。

- Tracker 服务器及 DHT 网络结点列表：请参考后文所讲的网络类型详细说明。
- 选项：只要保持默认设置，不用修改。不选择私有种子，而直接勾选其他的几个设置，如添加 DHT 到 Tracker 列表、文件按分块大小对齐、长效种子查找以及计算 ED2K 链接等。

注意：至于为什么要勾选这几个选项，读者可自行查阅 BitComet 官方帮助文档的相关说明。这里并不是本文讲解的重点，所以就不展开叙述了。

- 生成：保持“生成 torrent 文件”选中状态，单击浏览选择要制作出的 torrent 文件的保存路径。单击该窗口下方的“确定”按钮，就可以制作出一个 torrent 文件了。

注意：制作完 Torrent 文件后，要想将这个资源发布出去，还需要向 Tracker 注册。也就是说发布 torrent 文件前不要忘记了用 BitComet 打开它，自己作为一个种子 Peer，否则整个 BT 网络中就只有种子文件而无法进行下载了。

2．网络类型详细说明

（1）关于 DHT 网络和 Tracker 服务器

在 BitComet 中，通常都是同时使用 Tracker 服务器和公用 DHT 网络来寻找用户。列表框中可以填写传统 Tracker 服务器地址或者 DHT 网络结点地址，也可以只填其中一种，也可以什么都不填。这样，可以最大程度地保证 Torrent 文件的有效性。

在列表框中可以填写一或两个普通的 Tracker。这样既可以让使用其他客户端或 BitComet 旧版的用户下载，也可以让新版 BitComet 用户体验到 DHT 网络的稳定性。DHT 网络结点地址一般不需要填写，如果要填写，请参考下面将要讲述的有关 Tracker 服务器及 DHT 网络结点列表填写示例。填写这个列表，一般有 4 种情况：

- 什么都不填代表仅仅使用 DHT 网络，自动连接结点。
- 只填写 DHT 网络结点代表仅仅使用 DHT 网络，而且默认连接这几个结点，比如 node://router.bitcomet.net:554/。
- 只填写 Tracker 代表同时使用 Tracker 和 DHT 网络，自动连接结点。比如 udp://tracker.bitcomet.net:8080/announce
- 同时填写 Tracker 和 DHT 网络结点代表同时使用 Tracker 和 DHT 网络，而且默认连接这几个结点，比如：udp://tracker.bitcomet.net:8080/announce 和 node://router.bitcomet.net:554/。

（2）不使用公用 DHT 网络（优先使用 Tracker 服务器）

一般情况下仅仅使用 Tracker 服务器来寻找用户，列表框中需要填写传统的 Tracker 服务器地址。这个选项等同于以前版本的 BitComet 的 Torrent 制作方法。新版 BitComet 对于这种文件将首先尝试连接 Tracker。在长时间不能连接（时间超过了选项中的添加备用 Tracker 的时间）的时候，再添加备用 Tracker，并开始在公用 DHT 网络中寻找用户。这样就可以在 Tracker 服务器出问题的情况下依然能够下载。

（3）仅从 Tracker 服务器获取用户信息（禁用 DHT 及用户来源交换）

禁止使用公用 DHT 网络以及用户来源交换来寻找用户，列表框中需要填写传统的 Tracker 服务器地址。这种情况适合在内部使用的 Tracker 网络中分发文件。

3．BitComet的多服务器规范

服务器分组。互相联通并且信息共享的服务器为一组，比如同一个服务器的多个端口。同一组的服务器只要有一个连通了就不需要继续尝试这一组中的其他服务器。Bitcomet 将按组同时连接所有服务器。用“|”分隔同组的服务器。

BitComet 多服务器书写示例

Torrent 中字段：[[A 组服务器 1]，[B 组服务器 1]，[C 组服务器 1]]在 BitComet 的任务属性的服务器中写作：

A 组服务器 1

B 组服务器 1

C 组服务器 1

Torrent 中字段：[[A 组服务器 1，A 组服务器 2，A 组服务器 3]]在 BitComet 的任务属性的服务器中写作：

A 组服务器 1|A 组服务器 2|A 组服务器 3

Torrent 中字段：[[A 组服务器 1，A 组服务器 2]，[B 组服务器 1]]在 BitComet 的任务属性的服务器中写作：

A 组服务器 1|A 组服务器 2

B 组服务器 1

DHT 网络结点是 DHT 网络中任意一个长期在线的 BitComet 新版客户端的地址，一般完全不需要填写。如果要填写请按照 node://host:port/或者 node://ip:port/的格式填写在 Tracker 列表中，比如 node://router.bittorrent.com:6881/或 node://router.bitcomet.net:554/。

按照以上的操作过程，并进行相应的设置以后，在发布者的选项卡里，填入一些种子发布的信息，就全部完成了.torrent 种子文件的制作了。

6.4.5　BitComet 选项设置详解

在使用 BitComet 的时候，打开 BitComet 的主界面后，在菜单上选择“工具”|“选项”命令，就能进入关于 BitComet 选项设置的主界面。控制整个 BT 下载的状态、行为、过程以及参数和外观等的设置都可以在这里完成。本文只对各主要的选项设置情况进行讲解，如网络设置、任务设置、高级设置等。

在具体设置的时候，Bitcomet 中包含数字的各选项框均可使用上、下箭头调整或直接输入数字进行调整。如果需要恢复默认设置，删除安装目录下的 bitcomet.xml 文件后重新启动 BitComet 程序即可。

1．网络设置

打开选项设置的对话框后，出现如图 6.28 所示的界面，在这个界面的左侧显示了所有可设置的选项列表，第一项就是网络连接。

网络连接，主要完成 BT 联网状态的一些参数设置，其详细情况如下。

（1）连接设置，控制 BT 的下载和上传速度

- 全局最大下载速率：控制最大下载速度，默认为无限制，最低为 1KB。

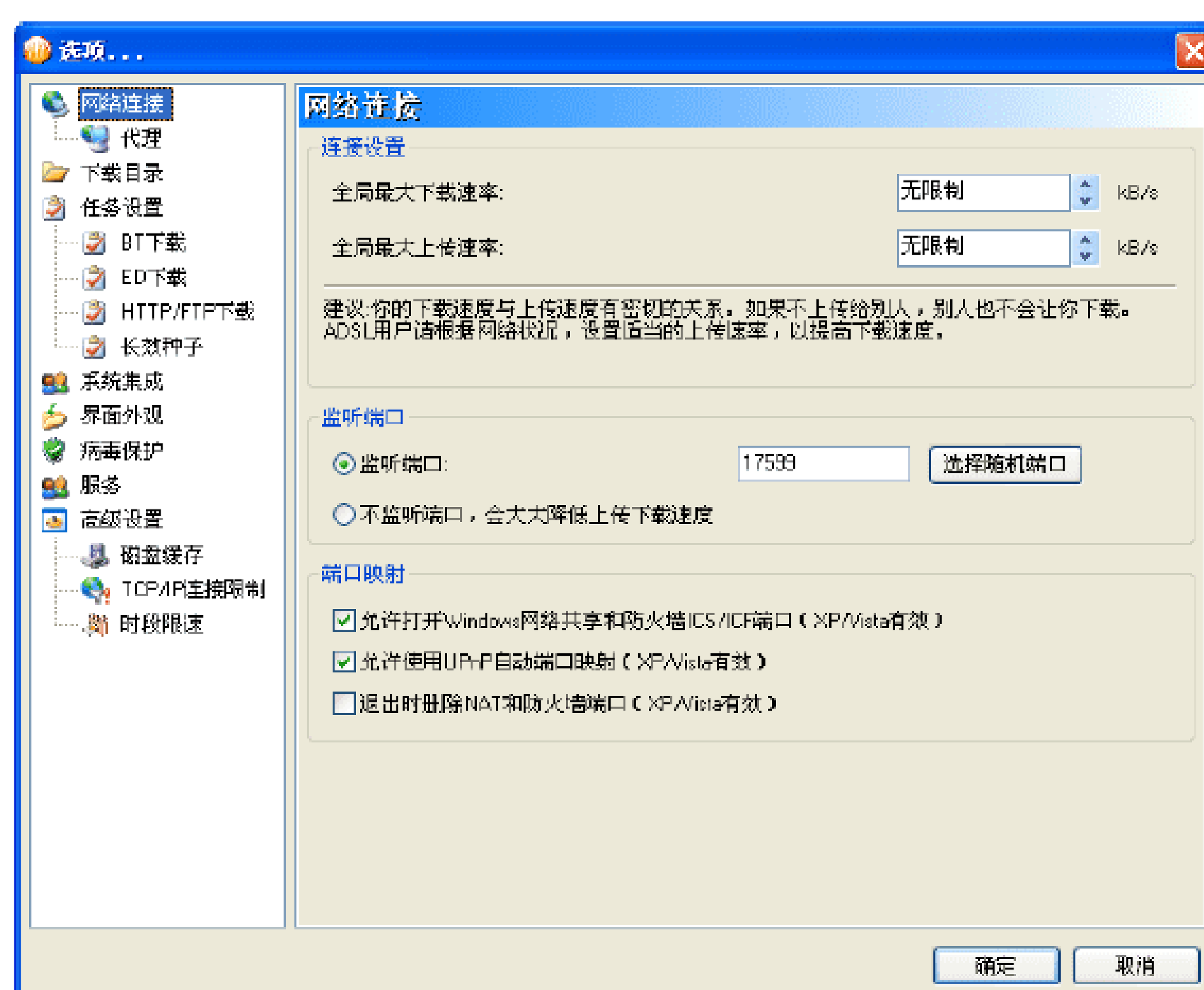


图 6.28　BitComet 选项设置界面

- ❑ 全局最大上传速率：控制最大上传速度，默认为无限制，最低为 10KB。

BT 下载的原则是上传越多下载越快，普通宽带用户使用默认无限制设置即可。ADSL 用户由于上传过大会影响下载，建议适当限制上传速度，一般为最大上传速度的 80%。在设置此项的时候，若前提已设置了“时段限速”，则连接设置失效。

（2）监听端口，BT 下载时对外监听的端口号

- ❑ 监听端口，选择随机端口：默认值为随机产生。

和原始 BT 软件不同，BitComet 只需要 1 个端口即可对应多个任务，通常不必修改默认设置。一般用什么端口都没有区别，如果已经在路由器上设置了端口映射，则在选项框中输入相应的端口即可

（3）端口映射，进行端口映射，用于防火墙和 NAT 的穿透

- ❑ 允许打开 Windows 网络共享和防火墙 ICS/ICF 端口（XP/Vista 有效）：默认为选上。

如果使用了 Windows 的网络共享或防火墙，其默认设置可能并不对 BitComet 使用的端口开放，从而造成 BitComet 连不上服务器或 peer 等问题，选择此项可自动在设置里添加允许 BitComet 端口通过。

- ❑ 允许使用 UPnP 自动端口映射（XP/Vista 有效）：默认为选上。

对于某些网关设备支持自动端口映射（极少见）的用户来说，此项设置可以自动打开网关设备对 BitComet 监听端口的映射从而使用户获得远程连接。如果能修改网关设置的，推荐手动进行端口映射。

注意：某些情况下，选择此项会导致一些断网、死机之类的问题，确认此选项对自己无效的用户推荐不选此项。

- ❑ 退出时删除 NAT 和防火墙端口（XP/Vista 有效）：默认为不选。

仅对以上两项成功的用户有作用，遗留相关端口设置可能会造成一些安全隐患，只是可能而已。这样，BT 网络连接就设置完成。下面几个选项的设置相对简单，这里不再详解。

2．任务设置

下面再说一下任务设置。选择“任务设置”选项后，进入任务设置的界面，如图 6.29 所示。

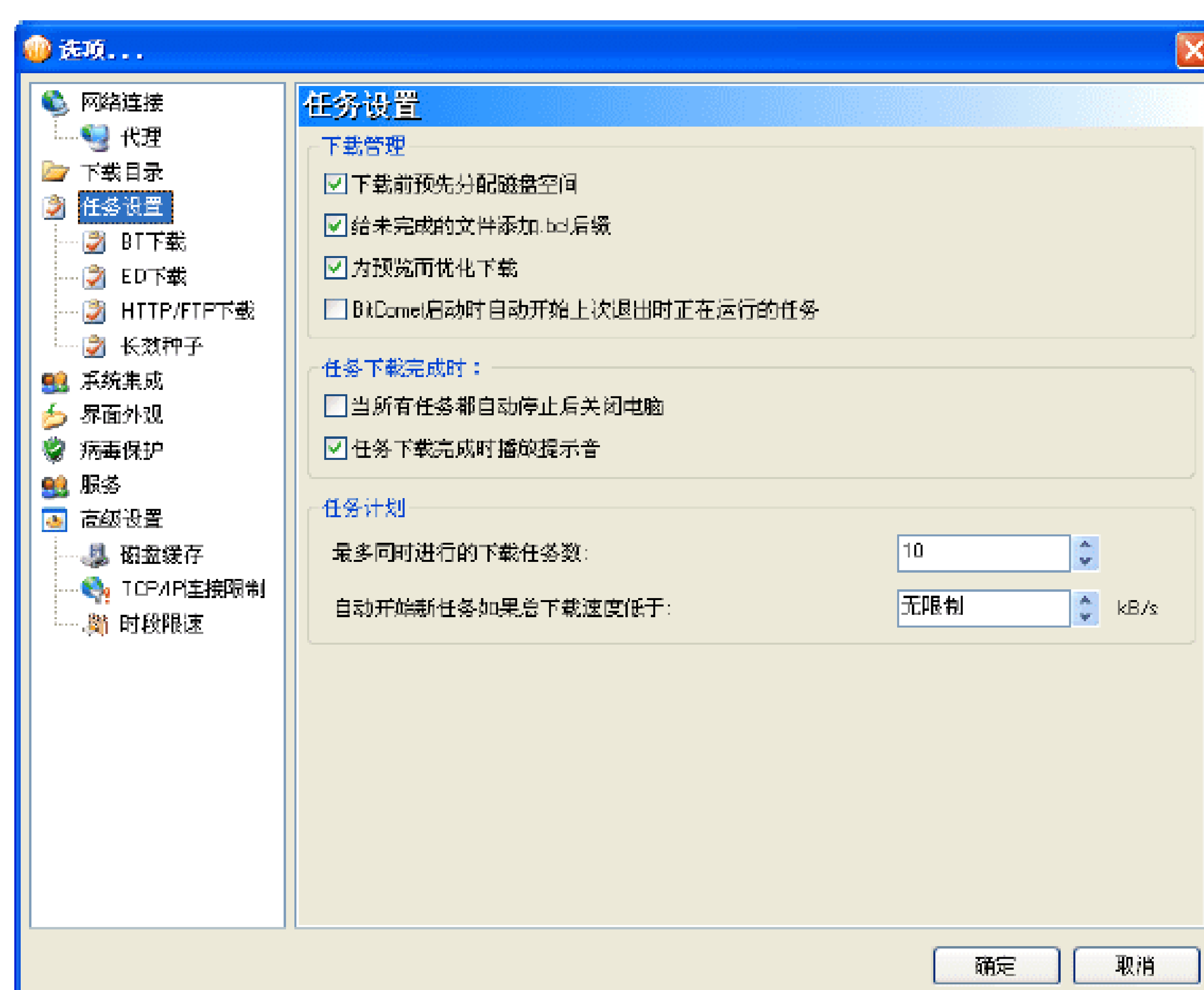


图 6.29　BitComet 选项中的任务设置对话框

任务设置，主要完成 BitComet 下载任务的一些配置，对 BT 下载、ED 下载、HTTP/FTP 下载等均起作用。

（1）下载管理

❑ 下载前预先分配磁盘空间：默认为选上。

勾选此项，可在硬盘上预先为下载文件分配应有的容量；否则将会在下载过程中动态地为文件分配空间。但是对于一些配置较低的用户，选择此项后，在下载大容量文件前的分配空间阶段，可能会有一段时间系统无响应。请根据实际状况设置此项。

❑ 给未完成的文件添加.bc!后缀：默认为选上。

勾选此项可自动为未完成文件添加!bc 的扩展名，任务完成后自动去除。但是个别时候会有文件下载不完全但已经可以使用的情况，而且因为一些用户不熟悉文件扩展名操作容易造成一些误会，建议没有特殊需要的用户不勾选此项。

❑ 为预览而优化下载：默认为选上。

勾选此项可以体验边下边看的功能。这个功能简单地说，就是一个 BT 文件没有完成 100%的下载也可以观看，这样，在下载的同时也可以观看。

❑ BitComet 启动时自动开始上次退出时正在运行的任务：默认为不选。

选择此项可以在 BC 启动时自动继续上次关闭 BC 时处于运行状态的任务，注意停止的任务是不会被自动运行的。

（2）任务下载完成时

❑ 当所有任务都自动停止后关闭电脑：默认为不选。

选择此项的同时必须在“BT 下载”选项卡中选择“自动停止任务”选项，才能生效。此功能很适合晚上挂机下载的朋友，当所有任务都符合条件自动停止后，会自动关闭电脑。

很实用的功能。

- ❑ 任务下载完成时播放提示音：默认为选上。

勾选此项让你在做其他操作，看不到任务列表中的任务进度时，知道有任务完成了。

（3）任务计划

- ❑ 最多同时进行的下载任务数：默认为 10。

🔔注意：此设置仅针对下载任务，上传任务不列入控制范围。超过设定值之后打开的下载任务将自动进入队列状态，排队等候开始（对队列中的任务再次选择开始，可突破选项设置）。

- ❑ 自动开始新任务如果总下载速度低于：默认为无限制。

如果在一段时间内总下载速度小于设定值，则会自动将队列中的第一个任务变成下载状态。注意停止的任务不能被自动开始。

在任务设置里还有系列的子项设置，这里只讲一下 BT 下载。打开“BT 下载”的设置界面，如图 6.30 所示。

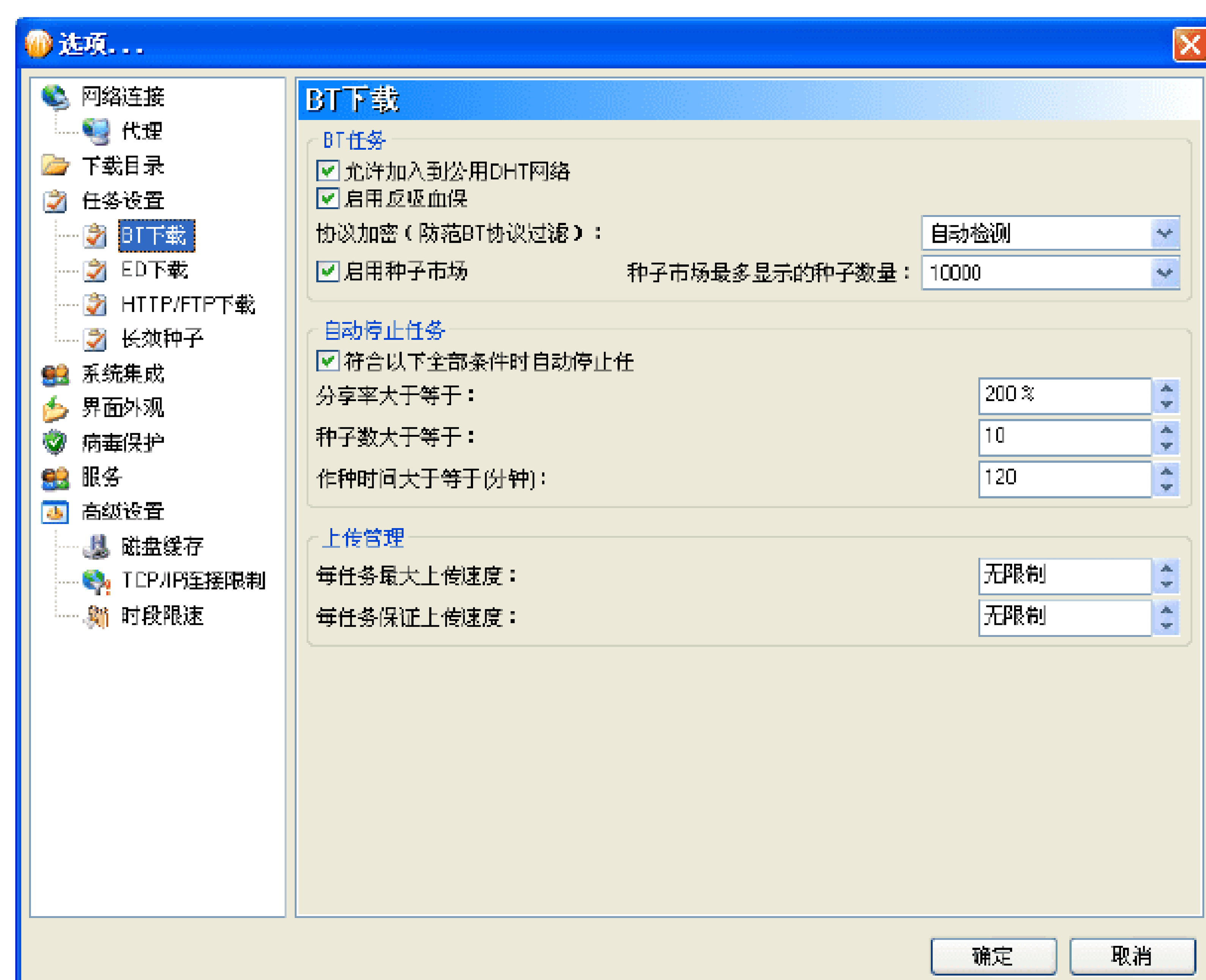


图 6.30　BT 下载设置界面

BT 下载的设置，主要有如下几项。

（1）BT 任务

- ❑ 允许加入公用 DHT 网络：默认为选上。

🔔注意：关于 DHT 网络，请参考本书 P2P 网络体系结构部分的知识。

- ❑ 启用反吸血保护：默认为选上。开启了反吸血保护，就会拒绝吸血 BT 客户端的连接，从而提高下载速度。
- ❑ 协议加密（防范 BT 协议过滤）：默认为自动检测。BitComet 对 BT 协议进行加密，可以更好地突破 ISP 的限制，使用户能够获得更多连接和更多数据。
- ❑ 启用种子市场：默认为选上。种子市场中最多显示的种子数量默认 1000。

（2）自动停止任务

符合以下全部条件时自动停止任务，默认为不选。

- ❑ 分享率大于等于：默认 200%，最低为 100%；
- ❑ 种子数量大于等于：默认 10，最低为 2；
- ❑ 作种时间大于等于（分钟）：默认 120，最低为 10。

无特殊需要，推荐保持以上 3 项的默认设定以体现共享精神，切记你的下载也正是来源自他人的上传。

（3）上传管理

- ❑ 每项任务最大上传速度：默认为无限制，最低为 3KB。

和全局速度限制不同，此项设置为每任务上传速度限制，且每任务速度之和若大于全局上传限制，则会取全局数为速度限制值。

- ❑ 每任务保证上传速度：默认为无限制，最低为 1KB。

BC 做种上传任务的优先级小于下载任务，导致某些情况下做种上传任务往往不能获得应有带宽。设置此项可使做种任务以保证速度以上的速度做种，避免做种但无上传速度的情况发生。

3．高级设置

BitComet 还有一个重要的设置，就是“高级设置”的选项，打开“高级设置”，出现如图 6.31 所示的界面。

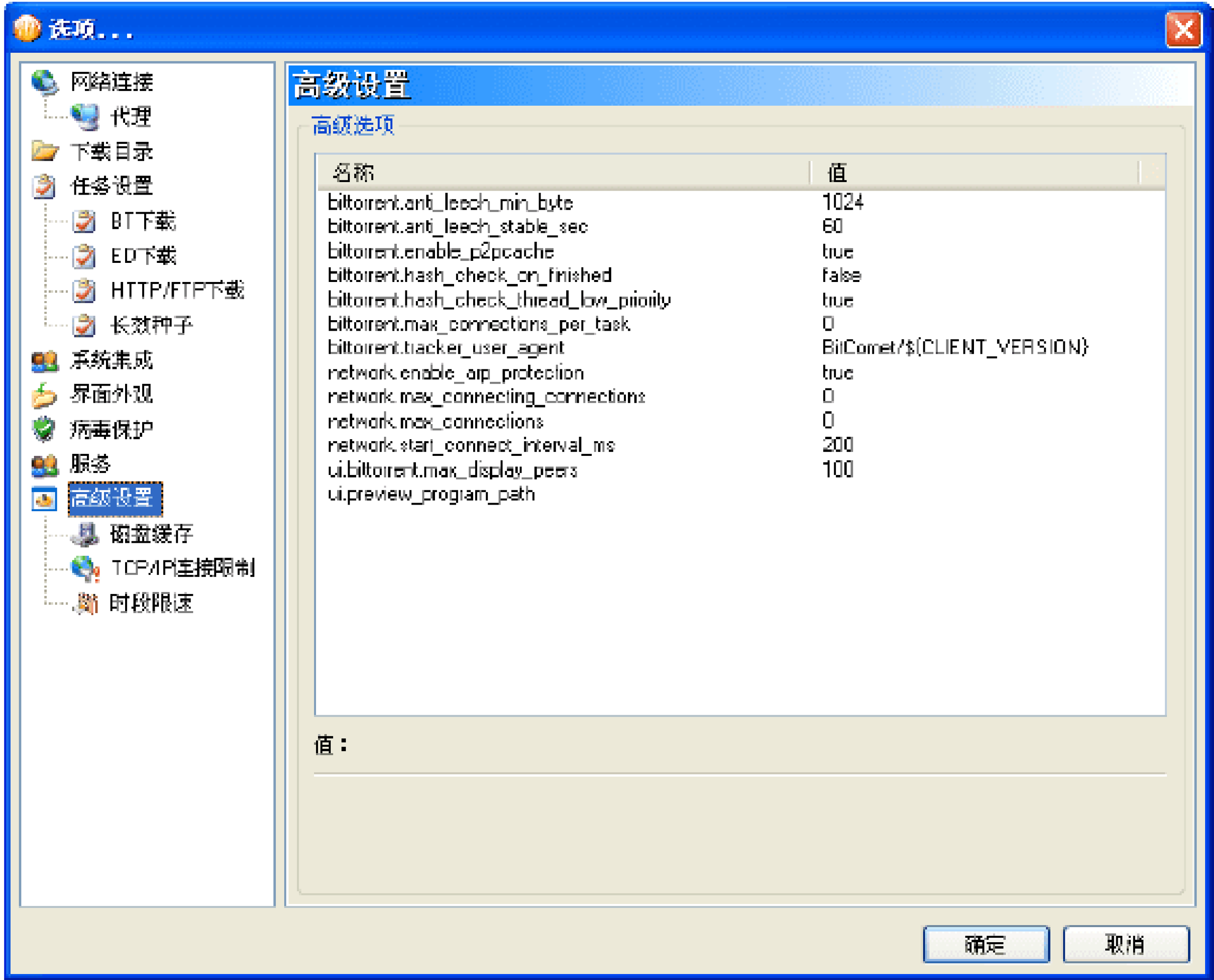


图 6.31　BitComet 选项高级设置界面

打开高级设置的界面后，会出现“高级选项”的参数列表，这个列表里各个参数的含义如下：

- ❑ bittorrent.anti_leech_min_byte：反吸血保护，对方在指定时间内需要上传的最少流量（byte）。
- ❑ bittorrent.anti_leech_min_stable_sec：反吸血保护，连接多长时间后开始检查流量（秒）。

- bittorrent.enable_p2pcache：是否允许任务下载时访问 P2P 缓存服务器（需要 ISP 支持），默认 true。
- bittorrent.hash_check_on_finished：下载完成时再次扫描文件确保完整性，默认 false。

注意：某些情况下显示为完成的任务不能使用，再次扫描完整性会显示并不是真正 100%完成。勾选此项可有效解决上述问题，但是扫描文件完整性时会占用大量系统资源，建议不经常出现上述问题的用户不选此项。

- bittorrent.hash_check_thread_low_priority：任务完整性检查线程设为低优先级，默认 true。
- bittorrent.max_connections_per_task：每项任务最大连接数，默认为无限制（0 为无限制）。该项是设置单个任务的最大连接数，一般用户不需要做修改。
- bittorrent.tracker_user_agent：连接 HTTP Tracker 时发送的用户代理，BitComet+版本号。
- network.enable_arp_protection：程序运行时防范局域网 ARP 欺骗攻击，默认 true。
- network.max_connecting_connections：最大尝试连接数，不是全局最大连接数，有很多用户把以上两个概念搞混淆了。
- network.max_connections：全局最大连接数，默认为无限制（0 为无限制）说明全局连接数里包含上传连接数和下载连接数，设置此项是另一种的限速办法，有需要的可根据实际情况调整。
- network.star_connect_interval_ms：连接发起间隔（毫秒），默认为 200。一般服务器端会对连接做出限制，因此客户端的调整并没有太大的意义。
- ui.bittorrent.max_display_peers：用户列表最多显示的用户数量，默认 100。
- ui.preview_program_path：设置视频文件预览时使用的播放器的路径。

向读者说明这些参数的意思，并不是告诉要怎么配置，而是通过这些参数与上文对 BitTorrent 协议规范的分析结合起来，就能更深入地理解 BT 的实现原理及其内部的工作方式。可以发现，这些参数与 BT 协议规范的要求是完全一致的。

在实际使用 BitComet 的时候，选项的高级设置值，非研究性使用 BT 的普通用户不建议修改。

磁盘缓存设置。

在高级选项的设置里，还有一个磁盘缓存的设置，打开“磁盘缓存”设置，进入图 6.32 所示的界面。

之所以有个磁盘缓存设置，主要是用于最大限度地保护用户的主机硬盘。在使用 BT 过程中，关于 BT 下载是否会影响硬盘寿命的问题一直存在争议。但毫无疑问，BT 下载过程中会同时上传和下载大量的文件，这必然会产生的大量读写操作，自然会大大增加硬盘的工作负担，而且上传下载的速度越高对硬盘的影响也就越大。为了将在使用 BT 的过程中对硬盘产生的不良影响降至最低，用于存储下 BT 上传下载文件的磁盘应该预留出足够的空间并经常整理磁盘碎片以避免增加硬盘的负担，同时还要设置磁盘缓存。

磁盘缓存的设置情况如下。

- 磁盘缓存最小值：默认为 6MB。

- ❑ 磁盘缓存最大值：默认为 50MB。

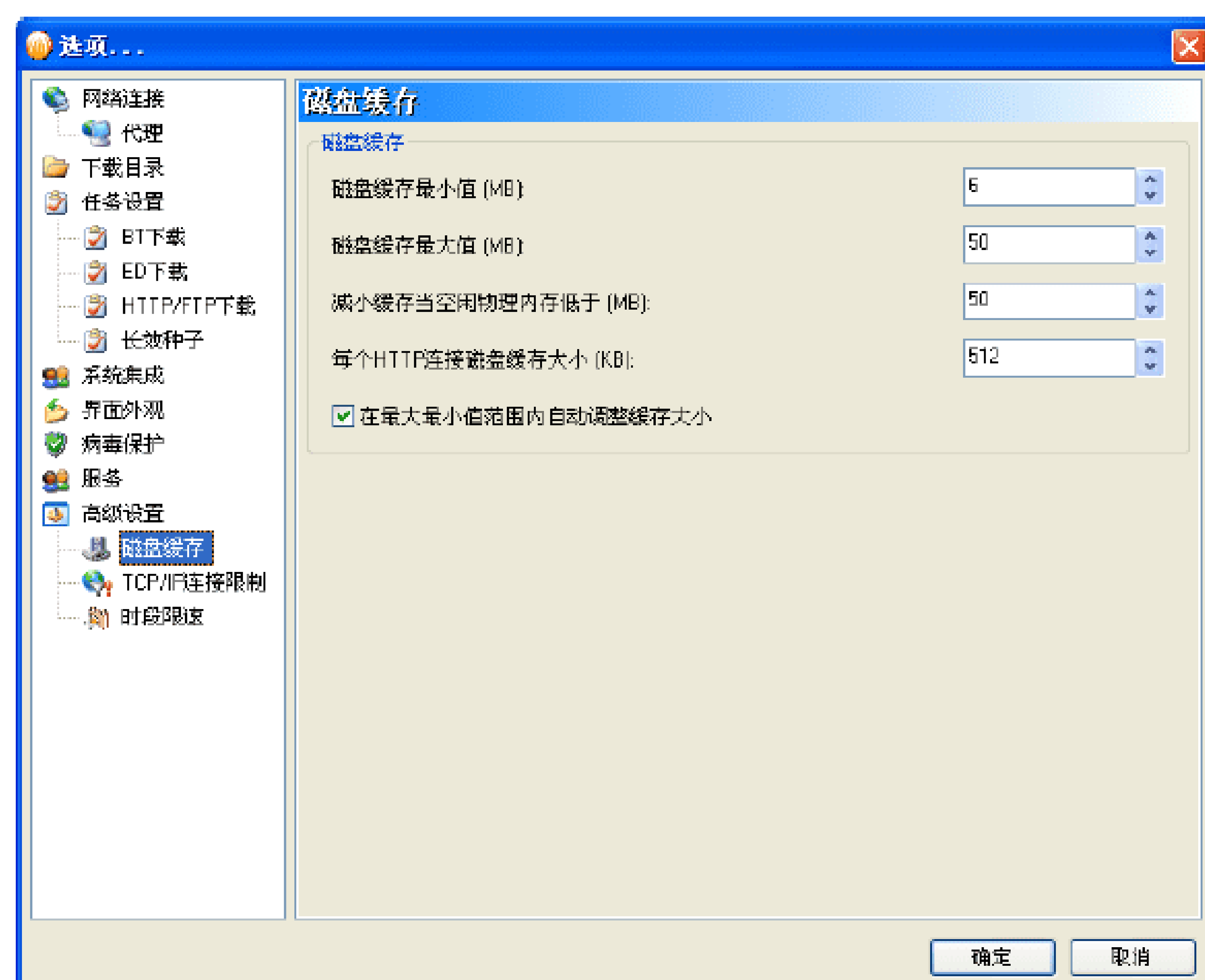


图 6.32　BitComet 选项磁盘缓存设置界面

合理设置磁盘缓存可在一定程度上降低硬盘损耗，BC 的智能缓存技术也可有效减少磁盘碎片的产生。在最大最小值范围内，BC 运行时的实际缓存使用量将取决于全局连接数等数据，用户可根据实际使用情况加大最大值（原则上缓存设置越大越好）。此项设置应该确保全局统计磁盘的读写命中率高于 90%为佳。

- ❑ 减小缓存当空闲物理内存低于（MB）：默认为 50MB。

一定的空闲物理内存将保证用户在使用 BC 下载的同时流畅进行其他操作，BC 将调整使用的缓存量以保证当前空闲物理内存大于设定值。

- ❑ 每个 HTTP 连接磁盘缓存大小（KB）：默认 512K。
- ❑ 在最大最小值范围内自动调整缓存大小：默认为选上。自动调整缓存可有效避免内存占用过大，同时减少磁盘碎片的产生。

注意： 这里磁盘缓存的最大值及最小值都是预估的，是在评估标准的硬件配置和下载速度的情况下进行估算的平均值。读者可根据自己的主机特征和下载需要进行配置，有些较智能的 BT 软件可以动态地自动配置最小和最大缓存。

至于其他选项的设置，都很简单，读者可自行摸索或查阅相关文档。

6.4.6　BitComet 的常见问题及解决方法

在使用 BitComet 的过程中，常会出现一个问题，也会产生疑问。下面就对一些常见的问题及解决方法进行简要说明。

1．BitComet和其他BT软件相比有何不同

BitComet 是少数的不使用 BT 官方 Python 内核的 BT 下载软件之一，而是用 C++完全独立重写的内核，曾经多次测试版检验并最终投入使用。BitComet 加入了 UPnP 自动端口

映射、内网互联等功能提升了内网用户的下载性能，至于下载速度，用户可自行试用比较。

2．BitComet安全吗

BitComet 基于 BT 协议进行开发的，在应用成系统的过程中有所改进（参考 BitComet 协议）。BT 是离散中心服务器（Tracker）型的 P2P 协议，目的在于高速分享大文件，而非匿名。下载/上传者的 IP 将被 Tracker 和其他下载/上传同一文件的用户获得（否则无法直接通信），正在下载的文件将被共享。但除此以外，用户机器上的其他文件都不会被泄露。

有时候刚打开 BitComet 的时候，防火墙就会报告 BitComet 正在连接某 IP 的 80 端口，是不是有木马呢？出现这种情况，主要是 BitComet 自身的自动提示版本更新或者广告图片更新。自动提示版本更新默认打开，可以在“选项”|“高级”选项里关闭。另外请按照本站的链接下载安装包，其完整性可以通过首页的 MD5 验证码来检查。关于网络防火墙及计算机安全方面的问题，请参考微软文档：保护您的 PC。

3．BitComet等下载速度时快时慢，不如HTTP或FTP下载速度稳定

BT 下载不同于传统下载，传统下载（HTTP、FTP）的文件位于服务器，只要服务器访问量在其设计范围内，下载的速度就算不快，也会比较稳定。而 BT 下载，服务器只提供 torrent 文件及 peers 的基本信息（地址和端口），被下载的文件则存在于 peers 的电脑中。这样就决定了 BT 下载的随机性——多数时候它都很快，但有些时候因为种子太少（只有一个种子时普遍很慢）、peers 普遍限速（这就是提倡“我为人人，人人为我”的道理）、peers 连接的随机性（或许在某次下载中，你连接到的 peers 都限速了或它没有优先向你传输数据）等原因所造成下载速度慢，那不是任何 BT 软件可以解决的，这就是 BT 下载的随机性。

当然，某些网络情况（例如受到内网、防火墙的影响）也会导致速度很慢。如果下载速度一直很慢，请考虑进行端口映射。

注意：关于端口映射的知识，请参考本书第 5 章“NAT 的穿透”部分知识。

4．如何提高BT下载速度

ADSL 或者其他内网上网方式下载速度本身就很慢，主要有以下几个原因：

- 处于内网，使其他下载者找不到你的主机，无法实现数据传输。
- UPNP 设备没有开启。
- 软件不适合你的机器，可换一个软件或升级一下版本试一试。
- 自身限速了，或者对方限速了。
- 要下载的文件没有种子了。
- 开启了防火墙。
- 机器长时间没有整理和优化了。

解决办法，以下解决方法分别对应上面的原因。

- 尝试进行端口映射。
- 开启 UPNP 设备。

- 使用 BitComet 下载软件，对外连接能力很强，0.54 版本更开通了内网互联功能。
- 不要过分地限速，有些 BT 系统上传与下载是对应成比例的，如果限制了上传的速度，下载速度也会随之下降。BT 下载完成之后，不要立即退出系统，最好还要能做一段时间种子，将自己的资源分享给其他的用户，因为 P2P 的精髓就是“人人为我，我为人人”。
- 需要耐心等待，如果长时间连不上就需要到求种区求种了，或者给其他下载完成的会员或种子发布者求种了。
- 给自己安装的防火墙做个端口映射。注意瑞星等防火墙会与 BT 冲突。
- 经常清除机器内的垃圾. 垃圾一般在 C 盘中的 Documents and Settings 的 temp 里和 Local Settings 里，一般都可删除，不清楚的文件请使用优化软件清理系统。

也可以试试看调整 BT 软件里的参数设置，调大端口范围，增加连接人数，限制上传速度等，但并不是绝对的。

注意： 出现以上的问题，解决方法不只是上文讲述的这几种，可能还有其他更好的解决方法。而所列出的这些解决方法中，也不一定就能解决问题，还需要读者在遇到问题时自己去思考、摸索。

5．BitComet显示UPnP成功，但是仍然没有“远程”

出现这个问题的可能原因是，在一个普通的网络环境中，网关或路由器一般由 2 个模块组成，网络防火墙、NAT。外部数据要进入内部网络，必须先通过网络防火墙，只有通过了网络防火墙，才能经由 NAT 转发给内网中的目标电脑。

BitComet 的设计遵照 UPnP 协议，可以在网关或路由器做自动端口映射，映射成功之后就会显示 UPnP 成功，但可能因为网关或路由器的防火墙设置问题，已经将外部的连接请求拦截，这个外部请求不能通过防火墙，更不用说到 NAT 模块后转发给内网中的目标电脑了。

所以，在这种情况下，虽然 BitComet 显示 UPnP 成功，但仍然没有“远程”。数据转发到本机端口了，但是由于本机防火墙不允许该端口的 TCP 连入从而被丢弃了。BitComet 可以自动配置 ICF 允许 BitComet 使用的那个端口，但是其他的网络防火墙软件，比如瑞星、天网等就需要自行配置了。

6．为什么在使用BT时，总是提示从服务器接受的数据损坏

出现此问题，可能的原因有 3 个。

（1）网络通信受外界干扰不够稳定，传输中的数据包受损，导致下载的部分数据出现错误。

（2）发送方出了错误，但以为是正确的数据。BitComet 显示的错误统计不只包括错误数据，也可能是通信协议的异常。

（3）与部分非官方 BT 客户端软件在扩展协议上不能完全兼容，可能导致下载数据错误。在 BT 客户端软件百家争鸣的今天，不少 BT 客户端软件除了遵循原有的 BT 协议，还进行了协议扩展。虽然所有的 BT 客户端软件都能够兼容 BT 协议，但扩展协议没有统一标准，所以会出现不同 BT 客户端软件之间不能够完全兼容、传输错误数据的现象。

解决方法：出现错误提示时，选择“常用”|“传送信息”|“外部 announce”命令，然后输入 http://btfans.3322.org:6969/announce，就可以连接了。

6.4.7　BitComet 快捷键及常用术语

BitComet 中，提供了很多快捷设置，可以提高用户的操作效率。在进行 BT 下载的时候，也会出现很多“术语”，这些术语对理解当前下载的整个进程和状态有很大的作用。下面简要地介绍一个常用操作的快捷操作和下载过程中一些术语的意思。

1．BitComet中几个常用快捷操作技巧

- 资源管理器中选中多个 torrent 文件，然后拖动到 BitComet 主窗口，可以一次加入很多 torrent。
- 资源管理器中将 torrent 拖动到 BitComet 悬浮窗，将会弹出下载确认框。
- 资源管理器中将某个文件或目录拖动到 BitComet 主窗口，将会弹出 torrent 制作窗口。

一些常用的快捷键如下。

- Ctrl + P：打开“选项”对话框；
- Ctrl +M：制作 Torrent 文件；
- Ctrl + O：打开 Torrent 文件；
- Ctrl + U：打开 BitComet 链接；
- Ctrl +N：新建 HTTP/FTP 任务。

2．BitComet中的一些常用术语的意思

（1）任务列表中什么是“健康度”？

BitComet 中健康度表示文件内容的分布情况。基本等同于其他 BT 客户端的“等效种子数”，但更简单实用。总地来说健康度超过 100%就可以下载，数字越大越好；健康度小于 100%就有可能下载不完或需要补种上传。当然所有这些都只考虑连上的 peer。其具体定义如下。

- 如果任务正在下载中：若网上的文件不全（可能下载不完），健康度就是网上存在的占尚需要的部分的百分比；若网上的文件全了（比如有种子），健康度就是需要下载的文件部分在网上的等效份数。
- 如果任务正在上传中：若网上的文件不全（除自己之外），健康度就是网上存在的部分能拼凑出来的占总的大小的百分比；若网上的文件全了（下载者之间理论上互相能补充直到完整），那就是总分布的等效份数（不包括自己）。

（2）任务列表中什么是“分享率”？

BitComet 中的分享率表示自己的分享程度，数字越大表示自己的贡献越大，人品越好。

- 下载任务：任务总上传量/任务总下载量。
- 做种任务：任务总上传量/文件总计大小。

（3）用户列表中什么是“远程”，什么是“本地”？

通俗地说，“本地”就是自己根据 IP 找到了其他人，“远程”就是其他人根据你的 IP

找到了你。如果没有监听端口，或者没有公网 IP，或者在网络防火墙后，或者网关没有端口映射，其他人是找不到你的或者找到了也连不上，也就没有远程连接，这样的用户通常被称作“内网”。任何人都可以找到并连上公网用户；内网用户只能主动去找别人；内网用户和内网用户之间一般互相找不到。

（4）用户列表中什么是“内网互联”（防火墙和 NAT 穿越）？

传统 BT 客户端中，下载同一个任务的公网用户可以帮助内网用户中转数据，但是内网用户互相不能连接上。BitComet 可以将不同内网的用户通过 UDP 互相连接实现内网互联。对内网用户来说这通常意味着更快的下载速度，因为可以连接上更多的用户。无需任何设置，BitComet 将会自动监测网络连接（自动判断是否处于 NAT 后或者没有端口映射），然后自动开启内网互联，从而加速内网下载。当然也可以在“选项”|“高级”选项中允许或禁止此功能。

（5）用户列表中的 ICic 表示什么？

BitComet 的 Peer 列表中的 ICic 分别表示如下（调试用，一般用户不必理会）：

- I：需要下载；
- C：不给上传；
- i：需要上传；
- c：不给下载。

学习 BT，不仅要知道它的原理和思想，更要知道它的应用，本节以 BitComet 为例详细讲解了 BitComet 的使用方法。其他的 BT 下载软件也都和 BitComet 具有类似的功能，读者要学会举一反三，要努力地把 BT 理论与应用两方面的问题都搞懂搞透。

6.5　BT 的影响及未来发展

基于 P2P 技术的 BT 发展及应用，改变了因特网的流量构成，将对等、共享的网络理念传遍互联网，对整个网络体系结构的发展、对网民的用网行为及网络资源的交流都产生了重要的影响。本节就简要说一下 BT 的影响及未来发展。

6.5.1　BT 的影响

BT 在互联网上的应用是把双刃剑，既有正面的影响也有不利的影响，它的影响是多方位、多领域的。

1．正面影响

可以让用户非常方便地下载和共享网络资源，节省了数据存储媒介和流通成本。例如，以线上游戏为例，有些线上游戏的线上更新（如魔兽世界）就是采用 BT 的技术。所以当每次有改版时，动辄数百 MB 的更新档，透过游戏厂商所提供的更新程式，以 BT 的方式进行下载分流。这与以往每次重大改版就必须重新压制光碟，或是等待单一下载点的下载方式相比，带来了另一种节省成本的经营模式。

时事新闻与外国电视无法收看，而通过使用 BT 方式下载节目既安全又方便，从而使

很多网民获得更多资讯。

2．负面影响

BT 下载方式目前引起社会的广泛讨论，尤其是在版权保护和不良信息传播方面。

利用 BT 免费发布版权内容肯定损害版权所有者的合法权益，但传播非收费性内容的好处有目共睹。争论的焦点是，是否应因此立法全面禁止 BT，并且对从事 BT 下载的人做出惩罚。

到目前为止，中国大陆和西欧国家等，对 BT 仍没有任何法律上的约束。而在中国香港，已经有人（绰号为古惑天皇）因为发布电影的种子而被海关拘捕。不光是 BT 技术，整个 P2P 技术都给版权滥用、不良信息传播带来了巨大的冲击。

3．其他影响

BT 可以最大限度地让全球用户充分共享和发布资源，能得到以前或其他途径无法获取的资讯和资源。

总之，BT 已经被广泛使用，它为许多并发的下载者提供成百兆的文件下载。但是，由于涉及版权等法律和道德领域的问题，BT 技术的进一步发展和应用尚需要规范化。

6.5.2　BT 的广泛应用

2006 年底发布的中国互联网统计报告显示，在中国 1.2 亿网民中有 27.18%的人使用过 BT 软件。而在国外，以 uTorrent 为例，据 P2P 领域新的调查数据表明，在 2008 年使用 uTorrent 的用户比去年增加了 2 倍多。在欧洲，BT 客户端是最流行的，安装率达到 11.6%，在美国用户量最少，但也有 5.1%的 PC 安装了 uTorrent。

2008 年 P2P 市场占有率统计图如图 6.33 所示。全球 uTorrent 用户使用情况统计图如图 6.34 所示。

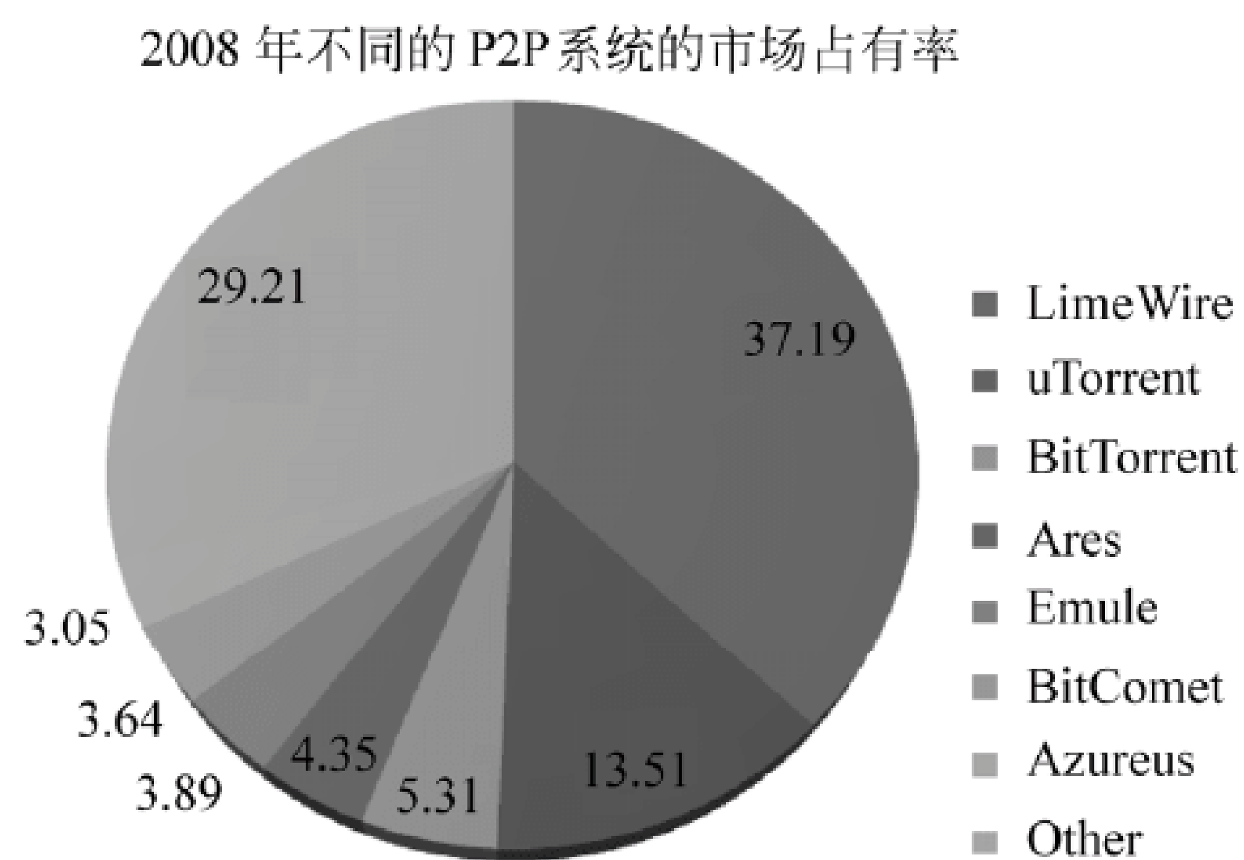


图 6.33　2008 年 P2P 市场占有率统计图

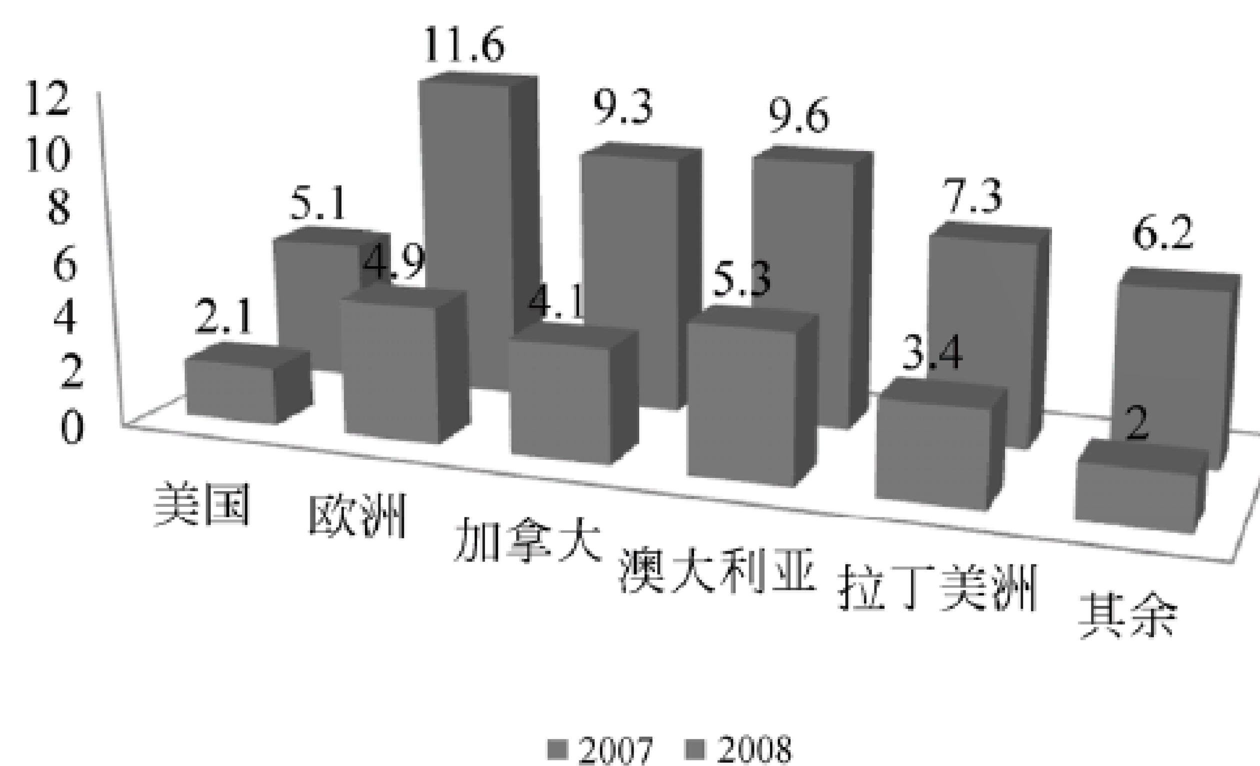


图 6.34　全球 uTorrent 用户使用情况统计图

由这两个统计图及相关数据可以清楚地看出，BT 技术在全球范围内都得到了广泛的应用，在全球的互联网用户中有着巨大的用户群，也有着相当潜力的市场前景。

6.5.3　BT 技术的发展

BT 技术在互联网生活的多方面有着广泛的应用。从它的进一步发展来看，除了上文所说的基于 BT 的文件共享与下载功能外，在安全、来源交换以及多媒体等方面也有深入发展的潜力。

1．在安全方面

BT 针对传统 P2P 模型中结点，以及资源缺乏统一管理和结点间交互的效率问题进行了改进。它引入追踪服务器，对结点和资源信息统一进行描述和发布，并引入传统计算机网络中 Web 服务器作为信息的发布平台，使 P2P 网络与传统网络间有机结合。从而能够将应用于传统计算机网络的管理策略、安全技术等引入 P2P 网络，极大地促进了 P2P 网络在管理和安全上的发展。

2．来源交换方面

Peer exange 技术是当前 BT 中研究的一个热点，就是在 BT 协议的基础上，将 Peer 结点源的发现，从 Tracker 服务转移到对待的结点上来。通过来源交换技术可以获取更多的源，进一步提高 BT 的下载速度。

3．在多媒体技术方面

由于 BT 模型提供文件高速分发，也逐渐开始应用于网络流媒体系统，例如视频点播、媒体广播系统等。

6.6　本 章 小 结

BT 是一种拥有自己独立协议用于文件分发的 P2P 软件，它以无结构的 P2P 模型为基础，结合 HTTP、TCP 协议等传统的网络技术，为大型文件的高效安全的分发提供了一个稳定的平台。

本章深入地讲解了 BT 的系统知识，对 BT 的工作原理、下载流程、协议规范等都进行了细致的分析，并以 BitComet 为例对 BT 客户端软件的使用、设置及常见问题也都进行详细的阐述，最后，简要说明了 BT 的应用和影响。本章内容力求让读者在全面把握 BT 知识的基础上，理解 BT 思想、掌握 BT 下载技术、了解 BT 系统模型和开发思路。

学习本章，要重点理解 BT 的工作原理和流理，理解 BT 协议规范，知道如何使用 BT 执行下载，将理论与实践结合起来学习。

第 7 章　基于 P2P 的 eMule 文件共享技术

同 BT 一样，eMule 也是 P2P 技术中最经典、最广泛的应用系统之一。作为一个成熟且不断发展壮大的文件共享系统，eMule 的许多特性和模型成为 P2P 应用开发中的典范。eMule 网络的自由性与匿名性，以及其强大的文件上传功能、超大文件的下载能力、丰富的资源拥有量，使得 eMule 已成为当前主流文件共享平台，从出现至今吸引了全球数千万的用户。学习 eMule 技术对掌握 P2P 技术的精髓、理解 P2P 技术的实现原理及进行 P2P 技术的应用开发都有着重要的意义。

本章就带领读者系统地学习整个 eMule 技术的知识体系，本章主要涉及的知识点如下。

- eMule 技术概述：概述性地讲解 eMule 的基础知识、来源、背景、历史及现状等，学会 eMule 最基础的、概念性的知识点。
- eMule 的原理：从网络结构、工作机制等方面讲述 eMule 的原理，重点掌握 eMule 的网络结构，掌握 eD2k、Kad 网络原理，掌握 eMule 的搜索下载原理
- eMule 协议分析：以 eMule 协议规范为基础重点讲解 eMule 协议的内容，详细分析 eMule 协议的核心知识点，重点掌握在 eMule 中客户端与服务器端及客户端之间的通信规范及过程。
- eMule 知识进阶：主要以 eMule 中的几个重要概念为出发点，结合 eMule 协议内容，让读者加深对 eMule 难点、重点的理解。
- eMule 使用：以实例的形式讲解了 eMule 的使用方法和一些设置步骤，读者要学会如何用 eMule 进行下载、上传及发布资源等操作。

7.1　eMule 文件共享技术概述

本节主要是概述性地讲解 eMule 的一些基础知识，从 eMule 的起源、来源、特点及版本等方面让读者对 eMule 文件共享技术有个初步的了解。

7.1.1　eMule 的起源与中文名称

eMule（电骡）的中文名称无论直译与否其实并无争议，因为官方已经给出了答案，请参考 eMule 官方网站说明：“eMule 来自一种叫做“骡子”的动物，提醒一下，就是那种有点像驴的家伙”。此处是“骡”而非“驴”，在中文搜索中使用“电驴”这个词是搜索不到 eMule 官方网站地址的，这也说明，官方认可的中文名称是“电骡”。

eMule，从单词的角度讲，这个单词可以拆分为 e +Mule。e 在英文中（尤其是计算机专业术语中）是“电子（electronic）”这一单词的简称，而 mule 一词的中文意思是骡子，

马骡；顽固的人，执拗的人；那么这两个词合起来，就是“电骡”的意思。这个词由它的前身 eDonkey 改进而来的，donkey 是驴，骡是驴和马的结合体，在身体素质上要好于纯种的驴和马。用 eMule 来命名正是来突出“eMule”的青出于蓝而胜于蓝的特点。

注意：electronic 一词，原意是用在微电子产品之上，在计算机英语中，也是常用的网络用语，常简写为 e，如 eMail、eBook 等。

2002 年 05 月 13 日，一个叫 Merkur 的人，由于他不满意当时的 eDonkey 2000 客户端并且坚信自己能做出更出色的 P2P 软件，于是便着手开发。他凝聚了一批原本在其他领域有出色发挥的程序员在他的周围，eMule 工程就此诞生。他的目标是将 eDonkey 的优点及精华保留下来，加入新的功能并使图形界面变得更好。现在 eMule 的最新版本是 0.49c。如图 7.1 所示的就是 eMule v0.49c 版的系统启动界面。

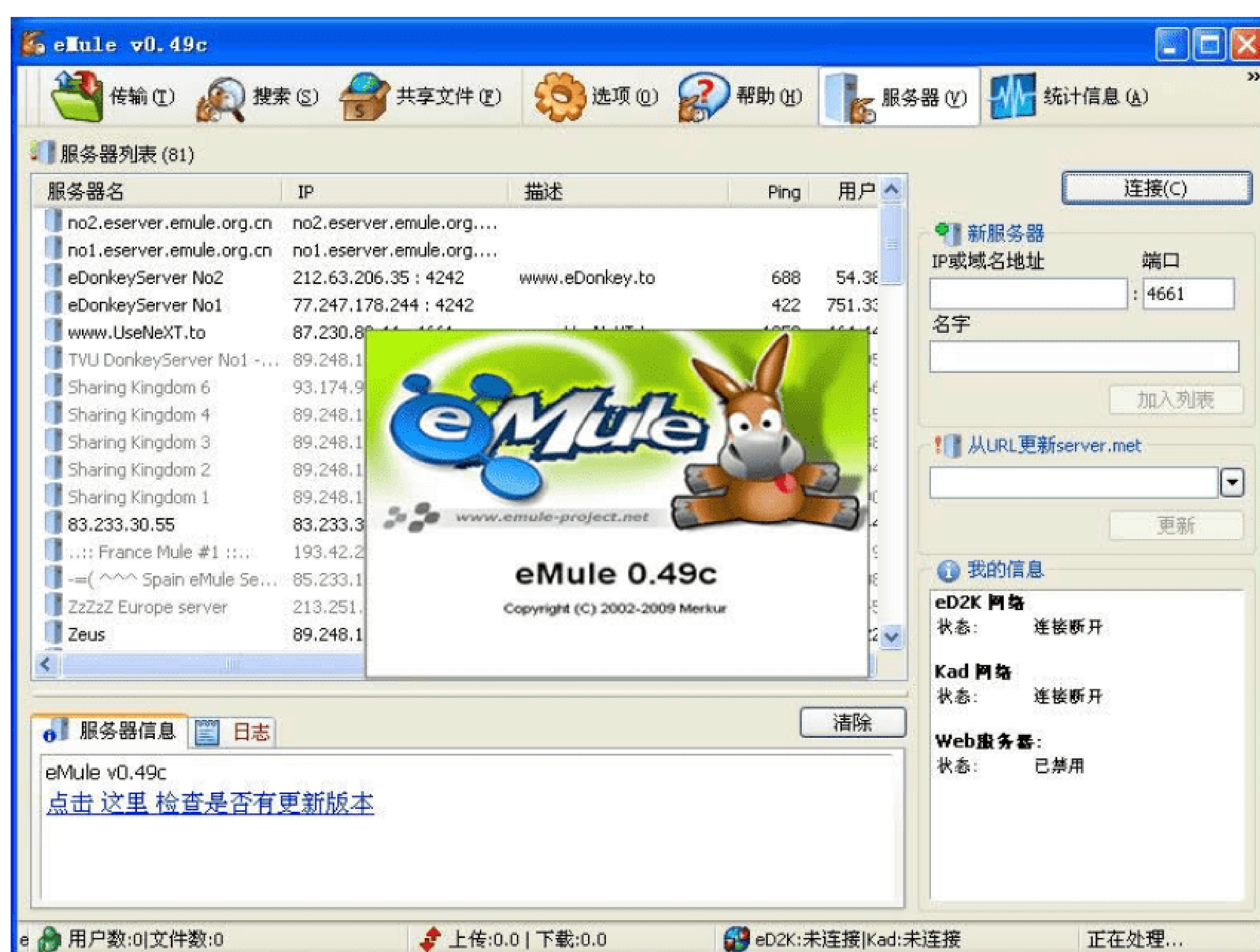


图 7.1　eMule v0.49c 版的启动界面图

注意：Merkur 是 eMule 的作者，他是一个德国人，本名叫 Hendrik.Breitkreuz（亨德里克.布雷特刘兹），他和很多来自世界各地的优秀程序员组成的开发团队，用自己宝贵的业余时间为 eMule-project 做出了贡献，而这一切完全是自愿而不求回报的。eMule 的开发团队曾说：“eMule 是完全免费的，它也决不包含广告软件、间谍和流氓软件。我们之所以创造 eMule 是为了快乐和知识，而不是为了金钱”。

7.1.2　从 eDonkey 说起

前面已经说过，eMule 是从 eDonkey 起源的，它的根是来源于 eDonkey 的，所以在详

细地讲解 eMule 之前，有必要了解一下 eDonkey。

注意：英语中 donkey 是指“驴”的意思，eDonkey 就是电驴。因为 eMule 来源于 eDonkey，所以，也有很多网友称 eMule 为“电驴”。尽管官方认可的 eMule 中文名字为“电骡”，但在现实生活中“电驴”也是 eMule 的意思。所以，本文中出现的“电骡”或“电驴”都指的是同一个意思，都是 eMule，不再另作解释。

1．eDonkey简介

eDonkey 是何物？能为我们做些什么呢？读者一定知道大名鼎鼎的 Napster 吧。eDonkey 2000 是由原先 Napster 中的几个技术人员设计的。

eDonkey，简称 ED，俗称“电驴”。eDonkey 是建立在 P2P 技术上的文件共享软件。众所周知，Napster 被起诉的主要原因是主机上的 MP3 文件供用户下载构成侵权。eDonkey 2000 的设计者吸取了 Napster 的教训，在文件共享的时候不作主机，只作索引。这就决定了 eDonkey 与传统文件共享的区别。

- 共享文件不是在集中的服务器上等待用户端来下载，而是分散在所有参与者的硬盘上。
- 所有参与者组成一个虚拟网络，每个用户端都可以从这个虚拟网络里的任何人的机器里下载文件，同时每个人也可以把自己的文件共享给任何人。
- 在 eDonkey 体系里有一些服务器，不过这些服务器不再存放文件，而是存放这些共享文件的目录或地址。
- 每个用户端从服务器处得到或搜索到共享文件的地址，然后自动从别的客户端处进行下载。参与的客户端越多，下载的速度越快。
- 在 eDonkey 的世界里，核心理念就是：“人人为我，我为人人”。

2．eDonkey原理

当你在搜索列表中选取了你要的文件并开始下载后，eDonkey 会记录下这个文件的大小、文件名及另一个叫做 Hash 的特殊值。说得更确切一些，是一个 MD4 的 Hash 值。

这个值是根据你要下载的文件本身的内容计算得来，它可以让你知道正在下载的文件是不是想要的。尤其是在文件的其他属性被更改之后例如文件名称等，这个值就更显得重要。

eDonkey 软件得到了这个信息后，会向所有添加的服务器发出请求，要求得到有相同 Hash 值的文件。而服务器则返回持有这个文件的用户信息。

这样客户端就可以直接和拥有那个文件的用户沟通，看是否可以从他那里下载所需的文件。

而 eDonkey 最出色的部分就在于你不是只在一个用户那里下载文件，而是同时从许多个用户那里下载文件。如果另一个用户只有你要的文件的一个小小片断，他也会自动地把这个片断分享给大家，而你就可以从这个用户的机器上下载这个片断。

当然你也一样。只要你得到了一个文件片断，系统就会把这个片断共享给大家。

3．eDonkey特点

eDonkey 的主要特点可描述如下。

- ❑ 不需要服务器来存放共享文件，节省了服务器架设、海量硬盘、网络带宽。
- ❑ eDonkey 软件搜寻速度快，可以搜寻在 eDonkey 2000 网络中所有共享出来的文件，允许传输任何格式的文件。
- ❑ 提供多路径获取文件，同时从多重地址下载同一个文件，以确保取得完整文件，所以可以确保你能拿到一首专辑里的所有歌曲或是一部电影的所有片段。
- ❑ 支持同步下载与上传，在下载时也可以进行上传共享的工作，以保持较高下载速度。
- ❑ 具有续传功能，若原先提供下载文件的服务器断线后，它会自动搜寻其他服务器，或借助在线的其他计算机继续下载。

4．从eDonkey到eMule

既然 eMule 是 eDonkey 的继承者，那么从 eDonkey 到 eMule 这个过程中，eMule 也必定继承了 eDonkey 所有的原理和特点。与 eDonkey 相比，eMule 还拥有了许多新的特性，总体而言，性能更为优越，功能更为强大。在了解了 eDonkey 的相关知识后，现在回到 eMule，再给 eMule 一个确切的定义。

eMule，是在第一代 P2P 网络 eDonkey 的基础上发展起来的，是建立在 P2P 技术之上的文件共享软件。

从 eMule 的发展与应用来看，eMule 已是世界上最大并且最可靠的点对点文档共享的客户端软件之一。作为一个完全免费并且开放源代码的 P2P 软件，利用它的卓越特性，不但可以与全世界的网友共同分享资源，更可以通过 VeryCD 网站，下载和发布最新的资源，充分享受自由共享的乐趣！

🔔注意：eMule 作为一个免费且开源的软件，并不是有规律地更新和升级的，一般是一周到三周一次。它的全球唯一官方网站是：http://www.eMule-project.net/。VeryCD 是 eMule 资源发布的一个 Web 网站。

7.1.3　eMule 与其他 P2P 软件相比的优点及特色

eMule 作为一款优秀的基于 P2P 技术的文件共享软件，与其他的 P2P 软件相比，有自身的很多优点和特色，主要有以下几个方面：

1．强大的搜索功能

eMule 客户端运行在由 eD2k、来源交换、Kad 共同组成的可靠的 eMule 网络结构中，具有强大的搜索功能。而且 eMule 也提供了一个大范围的搜索方式，包含服务器搜索（本地和全球）、基于 Web 搜索（Jigle 和 Filedonkey）及 Kad 网络等。客户端可使用多个不同的途径以搜索下载的资料源。

注意：Kad 并不是出现在 eMule 所有的版本中，在 eMule v 0.42 及后续版本中才真正使用 Kad。详细信息请参考 eMule 官方网站中关于 eMule 版本信息的说明。

2．完善的下载功能

使用 eMule 下载，每个下载的文件都会自动检查是否损坏以确保文件的正确性，其智能损坏控制有助于快速修复损坏的部分，eMule 的自动优先权及来源管理系统允许您一次下载许多个资源而无须监视它们。它的预览功能允许在下载完成之前查看、播放视频文件。用户可以自由地对下载文件进行组织和管理。

3．人性化的激励机制

在 eMule 中，加入了的排队机制和上传积分系统，通过这种方式，有助于激励人们共享并上传资源，以使自己更容易、更快速地下载自己想要的资源。真正做到了“人人为我，我为人人”的 P2P 网络理念。

4．免费与开源

eMule 遵循通用公共许可证（GNU General Public License）开发，任何组织和个人都可以永久免费安装使用，并且可以在符合 GPL 复制、散布与修改的条款与条件下发布 eMule DownCD 版、下载使用 eMule 的源代码。eMule 的官方版本也完全没有任何的广告软件、垃圾信息等，可以供人们免费使用、自由研究。

注意：GNU 计划是由 Richard Stallman 在 1983 年 9 月 27 日公开发起的。它的目标是创建一套完全自由的操作系统。

为保证 GNU 软件可以自由地“使用、复制、修改和发布”，所有 GNU 软件都包含一份在禁止其他人添加任何限制的情况下，授权所有权利给任何人的协议条款，GNU 通用公共许可证（GNU General Public License，GPL）。这个就是被称为“反版权”的概念。GNU 也针对不同场合，提供 GNU 宽通用公共许可证（GNU Lesser General Public License，LGPL）与 GNU 自由文档许可证（GNU Free Documentation License，GFDL）这两种协议条款。

GNU 的精髓就是，只要使软件在完整开源的情况下，尽可能使使用者得到自由发挥的空间，使软件得到更快更好的发展。

5．在线即时通信

eMule 在下载的同时，可以使用内建的 IRC 客户端，和全世界其他的共享者聊天。使用信息及好友系统，能传送信息到其他的客户端并可将他们加为好友。有好友上线的话，就能在好友列表中看到他（她）。

综上所述，eMule 与传统文件共享的区别是：共享文件不是在集中的服务器上等待用户端来下载，而是分散在所有参与者的硬盘上。所有参与者组成一个虚拟网络，每个用户端都可以从这个虚拟网络里的任何人的机器里下载文件，同时每个人也可以把自己的文件共享给任何人。

在 eMule 体系里有一些服务器，但是这些服务器不再存放文件，而是存放这些共享文件的目录或地址。每个用户端从服务器处得到或搜索到共享文件的地址，然后自动从别的客户端处进行下载，参与的客户端越多，下载的速度越快。

7.1.4　多样化的 eMule 版本

由于 eMule 是基于开源协议 GNU 通用公共许可证（GNU General Public License）发布的，于是便有了很多 eMule 修改版（eMule modifications）出现，这些修改版被通称为 eMule MOD's。

eMule MOD 往往比官方原版 eMule 有更多的功能和改进，比如显示地区旗帜、NAFC 带宽控制、有害 IP 过滤、强力上传、智能文件段共享、来源管理、缓存调整、快速启动、智能分类、反吸血客户端、老板健等，有着很多非常实用的功能，因此深受广大 eMule 爱好者的喜爱，甚至超过了官方原版的影响力。目前国际上比较流行的、优秀的常用 eMule 修改版本有下面几种。

注意： NAFC（Network Adapter Feedback Control）是网络适配器回馈控制的简称，以返回的信息对 eMule 进行速度限制。eMule 中使用 NAFC 功能可以根据网络流量动态调整 eMule 的上传和下载，使得所有应用程序都可以平稳地访问网络。开启此功能后，eMmule 的上传速率可以设置的较为接近最大上传能力，建议是最大上传能力的 80%～90%，当有其他程序需要访问网络时，NAFC 会自动调节网络流量，使 eMule 让出此部分网络流量。

1．eMule-Xtreme

Xtreme 是广受大家喜爱的 MOD，原作者为 x-man，在“电骡社区”里具有重要影响，系统截图如图 7.2 所示。



图 7.2　eMule-Xtreme 系统截图

基于此款 MOD 修改的版本较多，现在由 stulle 和 zz_fly 进行开发工作。在 eMule-Xtreme 中，有完善的 dlp 吸血骡屏蔽与 NAFC 网络状态实时监控、SLS 来源储存、文件强力发布等功能，深受“骡友”们的喜爱。较适合带宽较小的 ADSL 用户使用。

2．eMule ScarAngel

eMule ScarAngel 是 eMule 的另一个 MOD 版，俗称“天使骡，刀疤天使”。其作者是 stulle，ScarAngel 的系统截图如图 7.3 所示。



图 7.3　eMule ScarAngel 系统截图

eMule ScarAngel，基于 Xtreme，继承了 Xtreme 的所有特性，并加入了 MorphXT mod 的强力共享、增强的分类管理、传播条等更适合分享、文件发布的功能特性。比较适合低带宽的 HighID 用户和普通的 ADSL 用户及文件发布者使用。

3．eMule Mephisto

eMule Mephisto，俗称“魔鬼骡，恶魔骡”。作者是 stulle，基于 scarangel 开发而成，在 ScarAngel 的基础上，优化了上传系统，更适合将 eMule 作为上传的用户使用。系统截图如图 7.4 所示。

4．eMule MorphXT

eMule MorphXT，俗称“忍者骡”，由 leuk_he 等人开发，是另一款影响巨大的 MOD，系统截图如图 7.5 所示。



图 7.4　eMule Mephisto 系统截图



图 7.5　eMule MorphXT 系统截图

基于此款 MOD 修改的版本，成为 morphxt 系。morphxt 以稳定高效的上传性能见长，其彪悍的上传能力及强大的上传管理，受到了广大发布者的喜爱。

5．eMule MagicAngel

eMule MagicAngel，俗称“魔法天使骡”。原作者为 sfrqlxert，现由 Gomez82 接手，基于 MorphXT 开发而成，系统截图如图 7.6 所示。

eMule MagicAngel 除了继承了 MorphXT 强大的上传能力外，最大的改进是增加了完善的反吸血功能支持，无 NAFC 功能，适合网络能力较大的用户使用。

6．NeoMule

NeoMule，俗称“黑客帝国骡”。作者为 DavidXanatos，neo 的功能特性非常强大，是所有 MOD 中功能最全面的，设置也非常复杂，曾号称电骡之王，系统截图如图 7.7 所示。



图 7.6　eMule MagicAngel 的系统截图



图 7.7　NeoMule 系统截图

多共享文件时程序运行缓慢，资源占用较高。适合内网用户和多机同时开电骡使用（巫毒功能）。

注意：以上 eMule 的各种 MOD 版本都可以在网络找到，这里就不再提供下载链接。学完本章的知识以后，希望读者在 eMule 源代码的基础上，开发出自己的 eMule Mod。

7．其他MOD

其他 MOD，比如 X-Ray、StulleMule、SharkX、ZZUL、ZZUL-Plus、BastarD、AcKroNiC、EastShare、jMule 等也分别因为各自的特色而拥有自己的使用者。正是全世界热爱电骡 eMule 的程序员们在自己业余时间的努力，为我们贡献了如此丰富多彩的 MOD，形成了独特的电骡 eMule 文化。“eMule 是完全免费的，它也决不包含广告软件、间谍和流氓软件。我们之所以创造 eMule 是为了快乐和知识，而不是为了金钱。”，这正是 eMule 开发者们真实的心声。

7.1.5　永远不会出现在 eMule 中的特性

为了保证 eMule 可以长期、健康、有序地发展，eMule-project 小组订立了相关规则，

以保证某些不健全、不合理的特性永远不会出现在真正的 eMule 中，了解这些特性对了解 eMule 的特点和理念有重要的作用。这些特性如下。

注意：eMule-project，指 eMule 开源工程开发小组。

（1）同时连接多个服务器：产生这种额外的连接只是浪费带宽并加重服务器负担。

（2）网内多播：eMule 认为，网内多播是没有必要的，大部分 ISP 并不支持此特性。

（3）服务器轮询：在 eMule 中进行服务器轮询，只会浪费带宽并加重服务器负担。所以，在 eMule 中进行服务器的访问的时候不会进行轮询操作。

注意：服务器轮询是指在多服务器管理的系统中，目标主机会定期联系主服务器以交互有关执行的作业信息。这种联系主服务器的过程称为“服务器轮询”，该过程每隔一定的时间定期发生一次。

（4）网络匿踪：网络匿踪指的是在 eMule 网络中隐藏客户端的主机信息，通过匿名的方式与服务器或其他客户端进行交互。要实现这种方式只有将 eD2k 与 KAD 彻底推翻重新设计才能实现，这意味着放弃几乎所有老版本客户端。一般来说，网络匿踪（例如日本的 WINMX，SHARE，WINNY）会将多于 50%的流量浪费在隐藏客户端的踪迹，所以 eMule 中并不提倡，也没有此特性。

（5）Webcache：一种由 ISP 组建的旨在通过在网内建立缓存服务器以节省网外流量的机制。

eMule 的工作机制会引发对于 Webcache 的“滥用”导致网络过载，以及有可能因此造成隐私泄露问题。eMule-project 规定，在开发过程中不会为 eMule 添加 Webcache 的支持。

（6）支持任何其他 P2P 网络：eMule 是独特的，不会再有其他网络的支持（诸如 BT）。eMule 的设计者们所追求的就是一如既往矢志不渝地开发与改善 eD2k 和 Kad 协议，eMule 就是这样的高贵和独立，想有一个通用的支持所有 P2P 协议的系统，这种面面俱到的想法最终只会面面失败。因为 eMule 认为，同时支持更多的协议意味着更多的不稳定因素。

根据 eMule-project 的约定，违反以上这些特性的 eMule 大多是不被官方认可的，它们有的不遵守官方约定，有的纯粹就是吸血骡，这些 eMule 根本就是不入流的，不值一提。

注意：吸血骡，简单地说，就是一种只下载不上传的 eMule，像吸血鬼一样，完全不顾 eMule 的理念，只顾自己下载资源而丝毫不承担上传资源的义务，有这类特点的 eMule 都叫做吸血骡。

2009 年 5 月 13 日，eMule 电骡七岁了！在 eDonkey（电驴）被判决“死刑”4 年后，执行“死刑”3 年后，eMule（电骡）这个未经授权的仿制者迎来了自己七岁的生日。电骡的拥护者们感谢 Hendrik.Breitkreuz，因为他为我们带来了 eMule，他是电骡之父。感谢那些偏执的程序员，因为他们用自己的业余时间来维持着 eMule 的成长却从来不求任何回报。正是由于无数的默默奉献者们才有 eMule 成功的今天，希望本书的读者将来也能成为其中的一员。

7.2　eMule 文件共享系统的原理

eMule 是建立在点对点（peer to peer）技术上，以 eDonkey 协议为基础的文件共享软件，在网络结构上，它类似于 Napster，是一个有强大中心索引服务器的网络。eMule 网络由上百个 eMule 服务器和数以百万计的 eMule 客户端组成，eMule 在工作过程中主要完成 eMule 文件的搜索及下载。所以，理解 eMule 的工作原理，就需要理解 eMule 网络、理解 eMule 文件的搜索方法及下载机制。

7.2.1　eMule 网络结构

eMule 网络由 eDonkey 2000 覆盖网络（简称 eD2k 网络）和 Kademlia 覆盖网络（简称 Kad 网络）组成。从结构上看，它是一个有着中心索引服务器的混合式 P2P 网络结构。

eMule 网络主要由两种覆盖网络构成，一种是 eDonkey 2000 网络，另一种是 Kademlia 网络。

注意：覆盖网络，简单说就是应用层网络，它是面向应用层的，不考虑或很少考虑网络层、物理层的问题，详细说来，覆盖网络是指建立在另一个网络上的网络。该网络中的结点可以看作通过虚拟或逻辑链路而连接起来的。虽然在底层有很多条物理链路，但是这些虚拟或逻辑链路都与路径一一对应。许多 P2P 网络就是覆盖网络，因为它运行在互连网的上层。覆盖网络允许对没有 IP 地址标识的目的主机路由信息，例如，Freenet 和 DHT（分布式哈希表）可以路由信息到一个存储特定文件的结点，而这个结点的 IP 地址事先并不知道。覆盖网络被认为是一条用来改善互连网路由的途径，例如通过 QOS（服务质量）保障来实现高质量的流媒体

eDonkey 2000（eD2k）网络是分块下载的双层无结构 P2P 网络，由服务器层和客户层组成。服务器提供文件索引信息和服务器列表，但并不传递实际的文件数据。

注意：eDonkey 简称 ED，2000 就是 2K 的意思。所以 eDonkey 2000 网络就是指 Ed20K 网络，它只是一种网络概念，类似于其他的 P2P 网络，如 Napster 网络、Freenet 网络等。

Kademlia 网络是基于异或度量的结构化 P2P 覆盖网络，也称为无服务器网络。每个 Kad 网络结点既是客户端，又是服务器，提供文件信息发布、存储和检索服务。

注意：在 2005 年 5 月著名的 BT 4.1.0 版实现基于 Kademlia 协议的 DHT 技术后，很快国内的 BitComet 和 BitSpirit 也实现了和 BT 兼容的 DHT 技术，实现 trackerless 下载方式。而在 eMule 中也很早就实现了基于 Kademlia 类似的技术（BT 中叫 DHT，eMule 中也叫 Kad，注意和本文简称 Kad 的区别），和 BT 软件使用的 Kad 技术的区别在于 key、value 和 node ID 的计算方法不同。

eMule 客户端按预配置的服务器列表和 Kad 结点列表加入到 eD2k 网络和 Kad 网络中，并从中获取网络服务，客户端用 TCP 连接到其他客户端进行上传和下载文件。一个客户端可以从几个不同的客户端中下载同一个文件的不同文件块，同时也上传其拥有的文件分块。进行 eMule 下载之前，首先需要从网络上获得 eD2k 文件链接，eD2k 文件链接中包含了文件名、文件大小、文件标识和根哈希值等信息。

7.2.2　eDonkey 2000 网络

eD2k 在整个 eMule 的知识体系中有着重要的地位，理解 eD2k 的网络结构及原理，对理解 eMule 的原理有重要的意义。

eDonkey 2000 Network（简称 eDonkey Network 或 eD2k），是一种档案分享网络，最初用于共享音乐、电影和软件。与多数文件共享网络一样，它是分布式的；文件基于点对点原理传输，而不是由中枢服务器提供。

eDonkey 客户端程序连接到 eD2k 网络中来共享文件。而 eDonkey 服务器作为一个通信中心，使用户在 eD2k 网络内查找文件。它的客户端和服务端可以工作于 Windows、Macintosh、Linux、UNIX 操作系统。任何人都可以作为服务器加入这个网络。由于服务器经常变化，客户端会经常更新它的服务器列表。

eDonkey 用混合 MD4 摘要算法检查来识别文件。这使 eD2k 网络可以将不同文件名的同一文件成功识别为一个文件，并使同一文件名的不同文件得以区分。对大于 9.28MB 的文件，它在下载完成前将其分割；这将加速大型文件的发送。为了便于文件搜索，一些 Web 站点对比较热门的文件建立 eD2k 链接。这些网站通常也提供热门服务器列表便于用户更新。

注意：MD4 是 Ronald L.Rivest 于 1990 年提出的单向散列函数。MD 表示消息摘要（Message Digest），该算法对任意的输入数据位流产生 128 位散列值。所谓单向散列函数，指一个函数 H（M），作用于一任意长度的数据 M，返回一固定长度的散列值 h。

2004 年，eD2k 网络超过 FastTrack，成为互联网上应用最普遍的文件共享网络。虽然每小时、每天数字都在变动，但据估计，在 2005 年中期，eD2k 网络上按平均水平，大约有两三百万用户通过 100 到 200 个服务器共享了 5 亿到 20 亿个文件。

注意：FastTrack 也是一种基于 P2P 协议的技术，主要应用在 Kazaa、Grokstei、Mesh 和 Morpheus 等应用程序中。在 2003 年时，FastTrack 是当时最流行的文件共享网络。

这种网络的一个问题就是它需要专用服务器以保证网络的运行。它依赖于乐于花费大量带宽、CPU 时间的用户来运行服务器。这些服务器会承受很大的负载，至少理论上它们更容易受到来自互联网的攻击。为了解决这种问题，最初发明 eDonkey 的人开发了一个 eDonkey 协议的“继承者”—Overnet。而 eMule 也自行开发了 Kademlia 网络，通常称为“Kad 网络”。这些协议将克服“服务器依赖”。

注意：关于 Kad 网络后文会讲到，这也是为什么说整个 eMule 网络由 eD2k 网络和 Kad 网络组成的。

现在最流行的 eD2k 客户端就是 eMule。

7.2.3　Kad 网络

Kad 是 Kademlia 的简称，eMule 的官方网站在 2004 年 2 月 27 日正式发布的 eMule v0.42b 中，Kad 开始正式内嵌为 eMule 的一个功能模块，可以说从这个版本开始，eMule 便开始支持 Kad 网络了。Kad 的网络拓扑结构如图 7.8 所示。

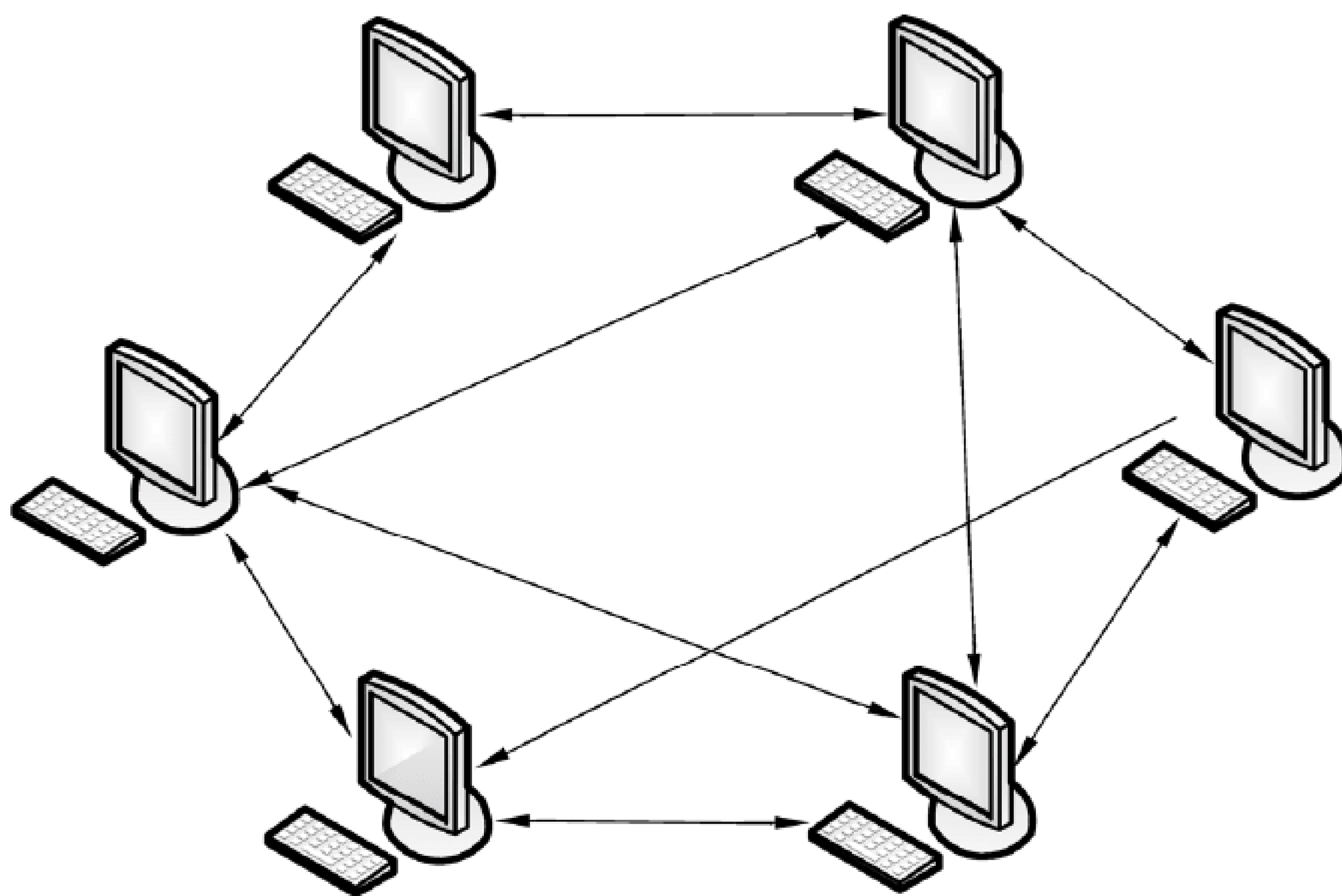


图 7.8　Kad 无中心服务器的网络拓扑结构图

Kad 的出现，结束了之前 eDonkey 时代，在 ED 圈里只存在着 eD2k 一种网络的模式，它通过新的协议开创并形成了自己的 Kad 网络，使之和 eD2k 网络并驾齐驱，而且它还完全支持两种网络，可以在两种网络之间通用。Kad 同样也属于开源的自由软件。它的程序和源代码可以在官方网站 http://www.eMule-project.net 上下载。

Kademlia 是 Petar Maymounkov 与 David Mazières 所设计的点对点（P2P）重叠网络，以达成非集中式的点对点（P2P）电脑网络。它规定了网络的结构规范了结点间的通信和交换信息的方式。简单地说，Kad 就是一种分布式哈希表（DHT）技术，不过和其他 DHT 实现技术比较，如 Chord、CAN、Pastry 等，Kad 通过独特的以异或算法（XOR）为距离度量基础，建立了一种全新的 DHT 拓扑结构，相比于其他算法，大大提高了路由查询速度。

注意：Kademlia 协议是美国纽约大学的 PetarP. Maymounkov 和 David Mazieres，在 2002 年发布的一项研究结果《Kademlia: A peerto -peer information system based onthe XOR metric》。

Kademlia 结点间使用传输通信协议 UDP 进行通信。Kademlia 结点以分布式哈希表（DHT 网络，distributed hash table）储存资料，通过既有的局域网 / 广域网（LAN / WAN）

建立一个新的虚拟网络或是重叠网络。每个网络结点都是以一组数字（“结点 ID”）来识别。这组数字不但作为识别之用，Kademlia 算法还会用来做其他用途。

一个想要加入网络的结点需要先启动。在这个阶段，这个结点需要知道另一个已在 Kademlia 网络内的结点 IP 地址（通过另一个使用者或储存的清单取得）。如果启动中的结点还不是网络的一部分，它便会计算一个尚未指定给其他结点的随机 ID 编号。这个 ID 会一直使用到离开网络为止。

Kademlia 算法是基于两结点间的“距离”来计算。这个距离是以两结点的 ID 进行异或运算，并将结果四舍五入至整数得到的。

注意：这个“距离”跟实际的地理环境无关，而是标明 ID 范围内的距离。因此一个德国的结点和一个澳大利亚的结点就有可能被称为“邻居”或“芳邻”。

Kademlia 内的信息都储存在称为“值（Value）”的数据内，每个数值都连接着一个“金钥（Key）”。

当搜寻某个 Key 时，算法会通过几个步骤，探查一圈整个网络，每个步骤都会更接近要搜寻的 Key，直到被连线的结点传回数值，或找不到更近的结点。网络的大小仅会稍微影响到进行搜寻时接触到的结点数目：假如目前网络的使用者突然增为两倍，那使用者结点大概只需要在搜寻时多查询一个结点，而不是两倍的结点量。

Kademlia 非集中式的结构提供了更大的优势，并很明显地增加了对拒绝服务阻断攻击的抵抗。即使一整系列的结点被拥塞，也不会对网络可用度造成太多影响，最后网络会通过绕过这些“洞”而自我修复。

7.2.4　Kad 网络的搜索机制

Kad 网络提供了帮助寻找结点以及记录结点的机制，也就是 Kad 网络的搜索机制。下面来讲一下这个机制的原理。

1．Kad网络中的几个基本概念

- UserID：Kad 网络中每个结点都有一个 160b 的 ID 值作为标志符。结点 ID 的生成，可以是根据特定信息 Hash 或者简单地随机生成（eMule 中 ID 为随机生成）。在 eMule 中是 128 位。
- 结点间距离：判断两个结点 x，y 的距离远近是基于数学上的异或的二进制运算，$d(x, y) = x \oplus y$，即对应位相同时结果为 0，不同时结果为 1。
- FileID：对文件内容计算哈希值，128 位，MD4 算法。
- k-bucket：也叫 K-桶，在 Kad 网络中，每个结点都保存和自己一定距离范围内的结点信息，用 k-bucket 结构来存储这些信息。k-bucket 的结构形式为（IP address , UDP port , Node ID）的数据列表。

K-桶的结构如图 7.9 所示。

K-桶内部信息存放位置是根据上次看到的时间顺序排列，最近看到的放在头部，最后看到的放在尾部。每个桶都有最大不超过 k 个的数据项，在 eMule 中 k=10。

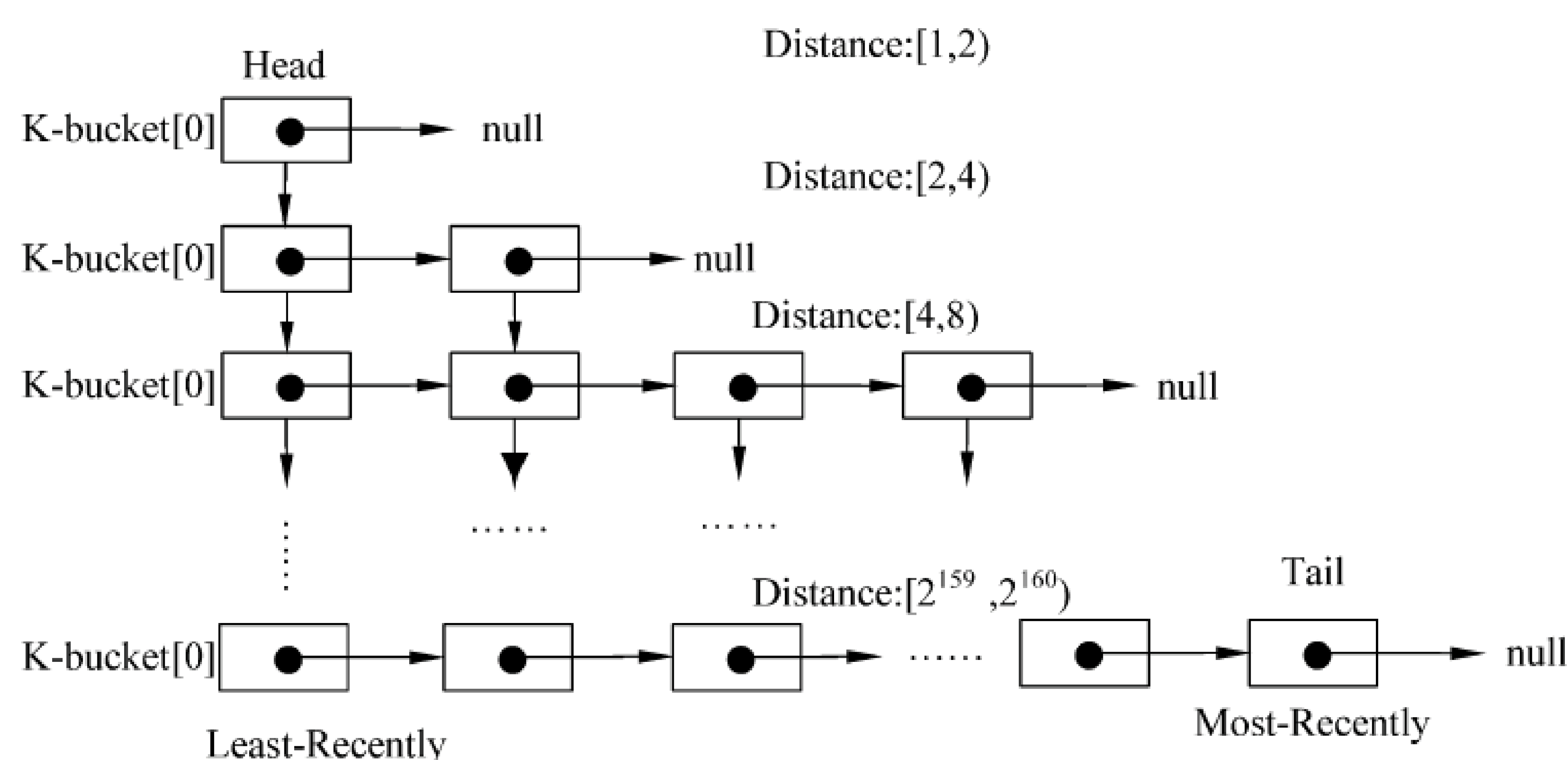


图 7.9　Kad 网络中 k-bucket 的结构

由于每个 K 桶覆盖距离的范围呈指数关系增长，这就形成了离自己近的结点信息多，离自己远的结点信息少，从而可以保证路由查询过程是收敛。也就是说，每个结点都对自己附近的情况非常了解，而随着距离的增大，了解的程度不断降低。经过证明，对于一个有 N 个结点的 Kad 网络，最多只需要经过 logN 步查询，就可以准确定位到目标结点。

注意：该结构会形成一个 tree 的形状，每一次接收到新的信息时，各个结点都必须更新 k-bucket 内的资料。通过 k-bucket 结构我们可以保证所有的结点状态都是新的，而且一定会知道这个结点在哪里。

2．Kad网络中的结点状态

在 Kad 网络中，所有结点都被当作一棵二叉树的叶子，并且每一个结点的位置都由其 ID 值的最短前缀唯一地确定。

对于任意一个结点，都可以把这棵二叉树分解为一系列连续的，不包含自己的子树。最高层的子树，由整棵树不包含自己的树的另一半组成；下一层子树由剩下部分不包含自己的一半组成；依次类推，直到分割完整棵树。图 7.10 就展示了结点 0011 如何进行子树的划分的。

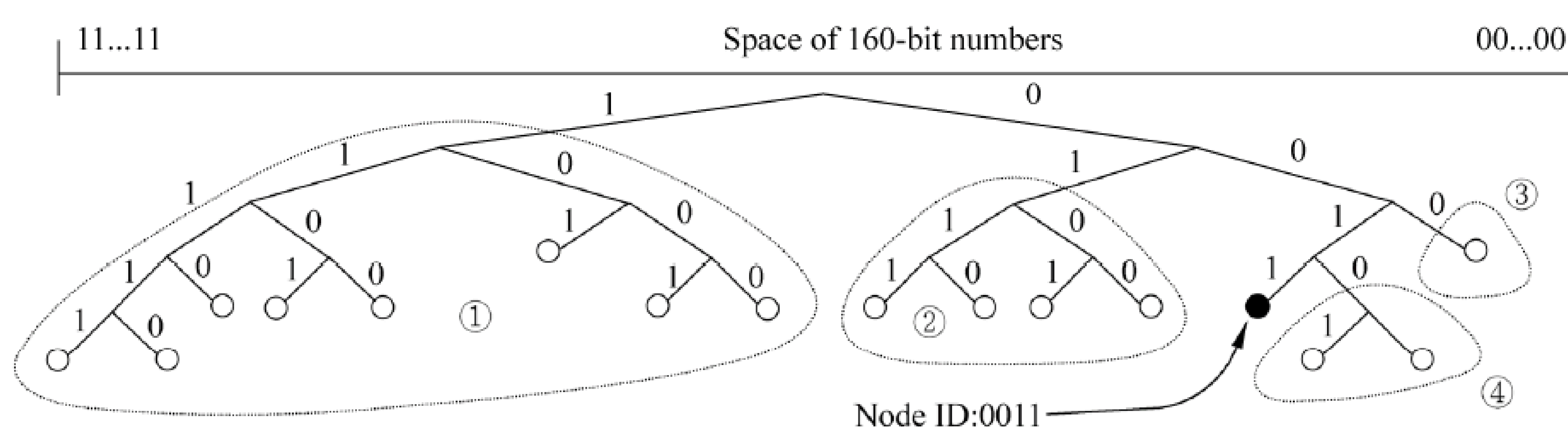


图 7.10　Kad 网络中结点 0011 的子树划分

图 7.10 中，虚线包含的部分就是各子树，由上到下各层的前缀分别为 0、01、000、0010。

Kad 协议确保每个结点知道其各子树的至少一个结点，只要这些子树非空。在这个前提下，每个结点都可以通过 ID 值来找到任何一个结点。这个路由的过程是通过所谓的 XOR（异或）距离得到的。

图 7.11 就演示了结点 0011 如何通过连续查询来找到结点 1110 的。结点 0011 通过在逐步底层的子树间不断学习并查询最佳结点，获得了越来越接近的结点，最终收敛到目标结点上。

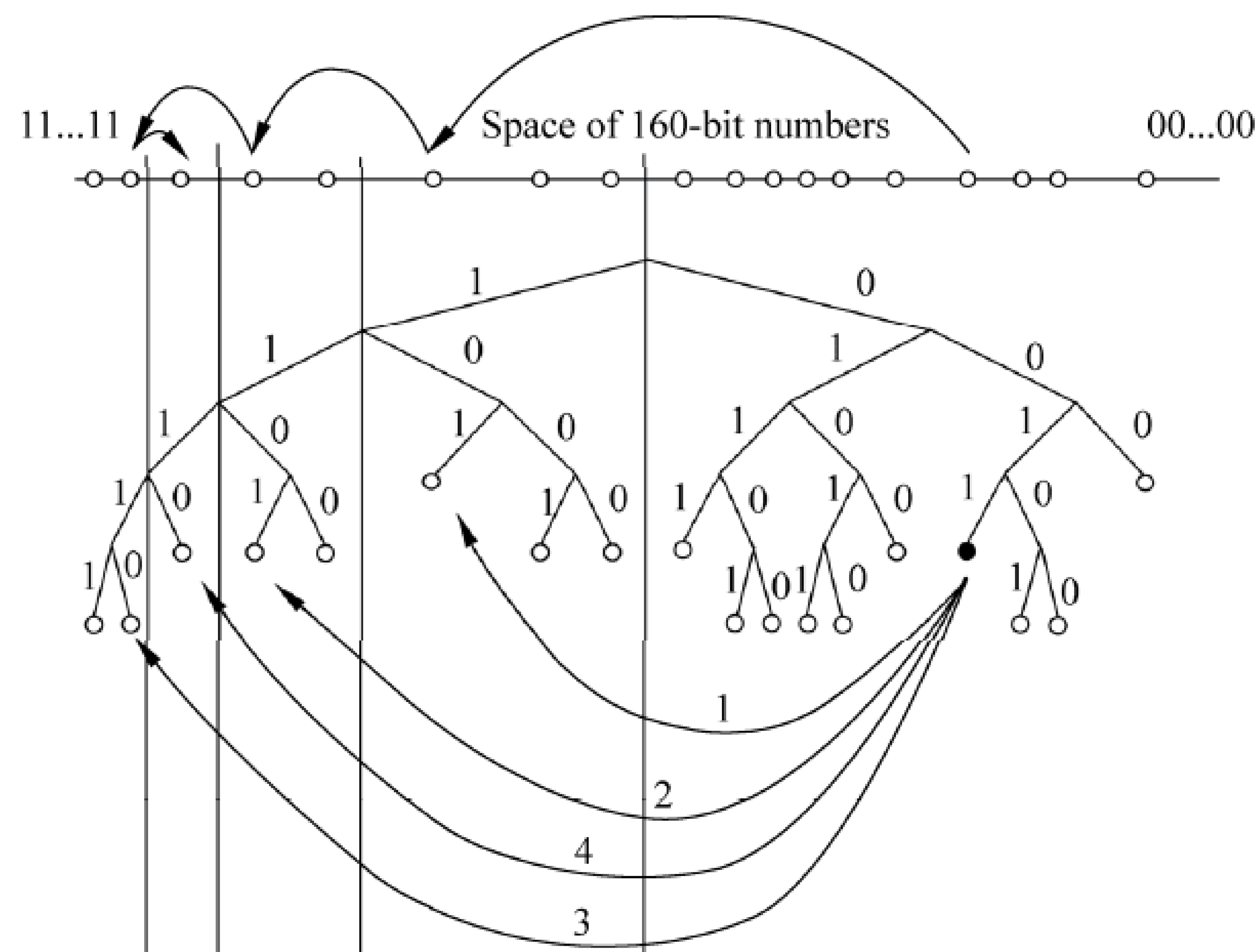


图 7.11　Kad 网络中通过 ID 值定位目标结点

需要说明的是，只有第一步查询的结点 101，是结点 0011 已经知道的，后面各步查询的结点，都是由上一步查询返回的更接近目标的结点，这是一个递归操作的过程。

3．Kad网络中的结点行为

Kademlia 协议包括 4 种远程操作（RPC），分别为 PING、STORE、FIND_NODE 和 FIND_VALUE，它们的含义如下。

- PING：操作的作用是探测一个结点，用以判断其是否仍然在线。对应于 eMule 中 PINGPING-PONGPONG 操作，即发送 KADEMLIA_HELLO_REQKADEMLIA_REQ 和 KADEMLIA_HELLO_RESKADEMLIA_RES 请求。
- STORE：操作的作用是通知一个结点存储一个<key，value>对。对应 eMule 中 publish 操作及 Store 操作。
- FIND_NODE：本操作的接收者返回它所知道的更接近目标 ID 的 K 个结点的（IP address、UDP port、Node ID）信息。
- FIND_VALUE：操作和 FIND_NODE 操作类似，不同的是它只需要返回一个结点的（IP address，UDP port，Node ID）ID 信息。

而当每一个指令被接收到后，每一个结点都会到 k-bucket 上搜寻，通过这样的结构，Kad 提供一个方便快速且可以被保证在 logN 次数下找到所需的结点。

4．Kad网络中的搜索过程

对架构在 Kad 网络中的 eMule 系统来说，它的搜索过程与 Kad 的路由机制有密切的关系。在 Kad 网络中，假如结点 x 要查找 ID 值为 t 的结点，Kad 按照如下递归操作步骤进行路由查找：

（1）计算到 t 的距离：d(x,y) = x？ y。

（2）从 x 的第[log d]个 K-桶中取出 α 个结点的信息（“[” “]”是取整符号），同时进行 FIND_NODE 操作。如果这个 K-桶中的信息少于 α 个，则从附近多个桶中选择距离最接近 d 的总共 α 个结点。

（3）对接收到查询操作的每个结点，如果发现自己就是 t，则回答自己是最接近 t 的；否则测量自己和 t 的距离，并从自己对应的 K-桶中选择 α 个结点的信息给 x。

（4）x 对新接收到的每个结点都再次执行 FIND_NODE 操作，此过程不断重复执行，直到每一个分支都有结点响应自己是最接近 t 的。

（5）通过上述查找操作，x 得到了 k 个最接近 t 的结点信息。

注意： 这里用“最接近”这个说法，是因为 ID 值为 t 的结点不一定存在网络中，也就是说 t 没有分配给任何一台电脑。

这里 α 也是为系统优化而设立的一个参数，就像 K 一样。在 BT 实现中，取值为 α = 3。当 α =1 时，查询过程就类似于 Chord 的逐跳查询过程，如图 7.12 所示。

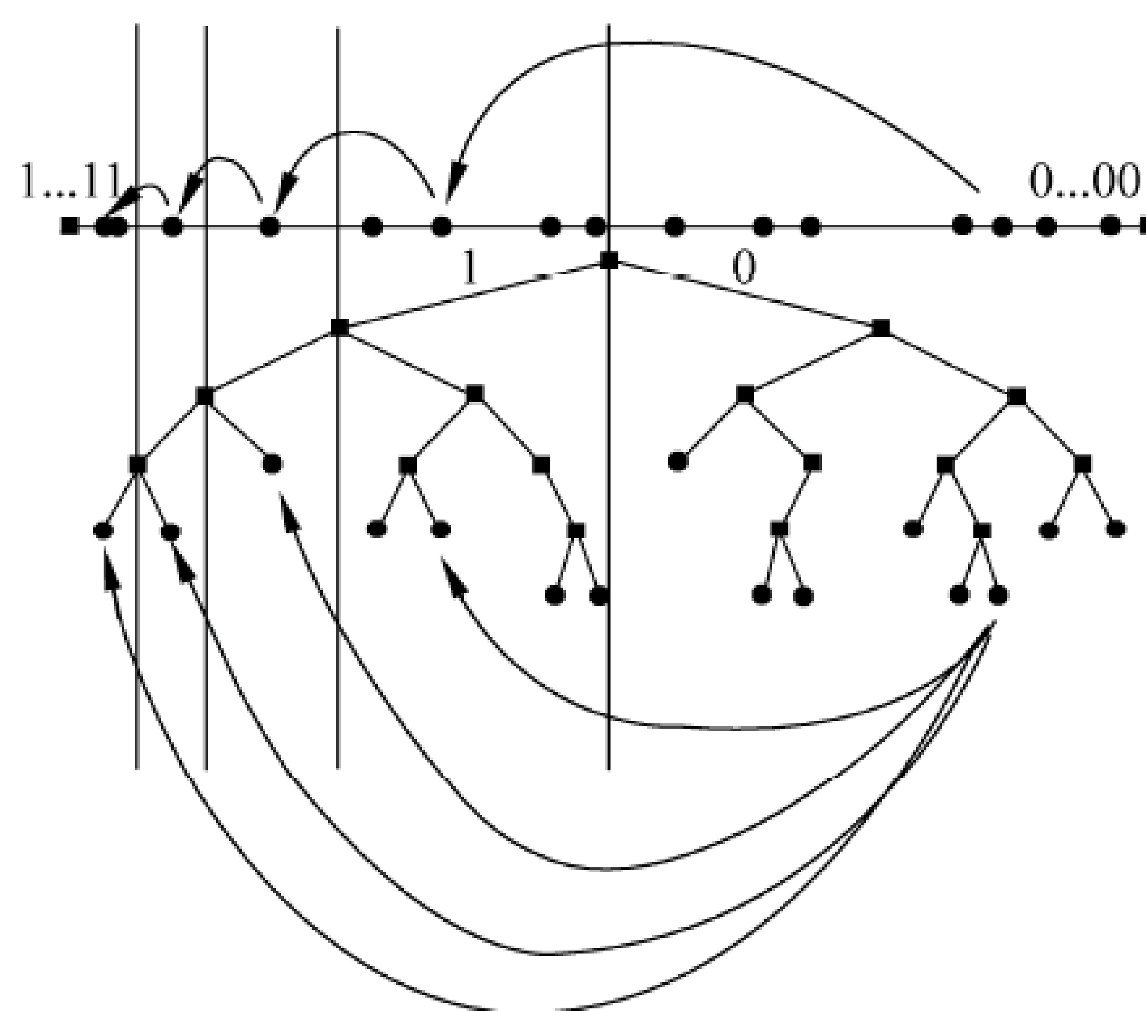


图 7.12　Kad 网络中的结点查询过程

在 eMule 系统中，利用 Kad 网络的路由机制来进行查找与以上的过程类似，客户端只负责处理一小部分搜索和查找源的工作。分配这些工作的时候，通过每个用户端唯一的 ID 和搜索文件的 hash 值之间的匹配来决定。

例如，“变形金刚.rmvb”这个文件由用户小王来负责（通过该文件的 hash 值来决定），那么任何用户在下载这个文件的时候都会告诉其他用户，小王有这个文件，其他用户去下载这个文件的时候也会询问小王，小王也会告诉他们谁正在共享这个文件，这样 Kad 查找源的工作就完成了。搜索时的方法也差不多，只不过是每个人负责一个关键字。

整个过程有点像在按照线索循序问路而找到正确方向，而不是在路上随便找人在问

路。而每个地方里的网络相关信息，则会随着电脑及文件的加入而持续更新。好处在于让你可以搜索整个网络，而不只是在某一地区。目前来讲，这个机制和算法是绝对领先而且非常优秀的。

在这个过程中，如何找到用户小王则是通过将用户 ID 异或的方式进行的。

- 两个 ID 的二进位异或值决定它们之间的逻辑距离，如 1100 距离 1101 要比距离 1001 近。
- 当一个用户加入 Kad 后，首先通过一个已知的用户找到一批用户的 ID 及 IP 地址和端口。
- 当该用户要寻找一个特定用户 A 的时候，该用户先询问几个已知的逻辑距离 A 较近的用户，如 B 用户、C 用户、D 用户。
- B、C、D 会告诉该用户他们知道的更加靠近的用户 ID 和 IP 地址及端口。
- 得到这些信息后，同理类推，这个用户最终就能找到 A。

所以寻找的次数会在 LogN 数量级，这里 N 代表询问的人数。

其实以上这种查找过程就是一种分布式散列的方法，基本上是对网络上某一特定时刻的文件进行快照（snapshot），然后将这些信息分散到整个网络里。为了找到特定的文件，搜索的内容先到达网络上的任何一台电脑上，然后这台电脑就会再将它转到另一台有更多文件信息的电脑上。第三台电脑可能拥有文件本身——或者也可能再继续转到其他有正确信息的电脑上。采用这种方法，通常只需要跳转两到三次，便可以轻松查找到所需的文件。

7.2.5　Kad 网络与 eD2k 网络的联系与区别

Kad 网络与 eD2k 网络，同是 eMule 网络的重要组成部分，它们既有区别也有联系，正是它们的优秀特点和互补的特性，共同成就了完善而健壮的 eMule 网络。

1. 在服务器上的区别

Kad 网络拓扑的最大特点在于它完全不需要服务器，传统的 eD2k 网络需要服务器支持作为中转和存储 hash 列表信息，Kad 可以不通过服务器同样完成 eD2k 网络的一切功能。作为客户端唯一要做的就是连线上网，然后打开 Kad。

2. ID上的区别

Kad 需要 UDP 端口的支持，之后 eMule 会自动按照客户端的要求，来判断它能否自由连线，然后同样也会分配给你一个 ID，这个过程与 eD2k 的高 ID 和低 ID 检查很像，不过这个 ID 所代表的意义不同于 eD2k 网络，它代表一个是否 freely 的状态。

（1）eD2k 网络里面，ID 的值是通过 IP 进行如下的算法计算得出的。（后文会有与 ID 值有关的程序算法实现）。

- 设 IP = A.B.C.D;
- 那么 ID number=A+256*B+256*256*C+256*256*256*D;
- low ID 的产生是由于我们的 ID 计算结果小于 16777216;
- 即 ID number=A+256*B+256*256*C+256*256*256*D<16777216，那么就规定此 ID number 为 LowID；反之就是 HighID。

（2）Kad 的 ID 计算原则并不是上面那样，它更关注我们是否是开放的、自由的。Kad 里的 ID 计算方法是这样的：

- ID number=256*256*256*A+256*256*B+256*C+D。

Kad 其实也有高低 ID 的分别。所以内网用户在使用的时候依旧无法达到内网用户完全穿透网络的效果，而就目前来看，还存在着 Kad 模块引入，导致占用系统资源会变大以及会突然产生 Memory Leak 的问题。对于内存的控制，目前 eMule 做的效果还是不好。

3．相同的目的

Kad 和 eD2k 网络有着完全不同的观念但是目的相同：都是搜索和寻找文件的源。

Kad 网络的主要的目标是做到不需要服务器和改善可量测性。相对于传统的 eD2k 服务器只能处理一定数量的使用者（在 eD2k 网络中，对每个服务器都有最大人数限制），而且如果服务器比较大连接人数过多，还会严重地拖垮网络。

总体比较而言，Kad 能够自我组织，并且自我调节最佳的使用者数量及他们的连接效果。因此，它更能使网络的损失达到最小。由于具备了以上所叙述的功能，Kad 也被称之为 Serverless network（无服务器网络）。虽然目前一直处于开发阶段（alpha stage），但它无可比拟的优势毫无疑问。

注意： 其实 Kad 本身有一个 nodes.dat 文件，也叫做结点文件，这里面存放了我们在 Kad 网络中的邻居结点，客户端用户都是通过这些结点来进入 Kad 网络的。Kad 的网络更像是 overnet 和 Kazaa 网络，有兴趣的读者可自行研究。

4．在eMule中使用Kad的好处

在 eMule 中打开 Kad，你会发现有以下几个明显的特点：

- 下载速度会加快；
- 下载文件的源会增加；
- 搜索的文件会增加。

以上三条对于 LowID 和经常下载源在国外的文件用户，效果就更为突出，特别对于在 eD2k 网络中只有几个源或者没有源的文件。在 Kad 网络中，一般都能找到源，所以说使用了 eMule 下载文件，基本上不会出现没有源的情况，无论多长时间，差别只是源的多少问题。

另外对于我们搜索的时候，如果采用 Kad 网络搜索，多数情况下找到的文件源会远远多于 eD2k 的全局搜索，这对于大家都是一个明智的选择。

7.2.6　eMule 的工作原理

了解了整个 eMule 网络的构成和特点，下面来说一下 eMule 系统的工作原理。eMule 建立于多点文件传输协议之上。一个 eMule 网络由服务器端和客户端两部分组成。服务器端是客户端连接的、为了搜索和查找可以下载用户的桥梁。所以整个 eMule 的工作都围绕 eMule 客户端与服务器之间、多个 eMule 客户端之间的交互进行的，在这个过程中重点解决好工作流程的控制和 eMule 的搜索及下载问题。

1．eMule的工作流程

在 eMule 的结构中，eMule 服务器列表像电话本一样排列，客户通过浏览它而获取需要文件的所有者的客户端信息，这是 eMule 工作的大的平台。在这个平台下 eMule 的工作流程有以下几步。

（1）eMule 开始工作起源于对 eMule 文件资源的搜索。

（2）当在搜索列表中选取了所要的文件并开始下载后，eMule 会记录下这个文件的大小、文件名及另一个叫做 hash 的特殊值。

（3）得到这些信息后，eMule 客户端会向所有添加的服务器发出请求，要求得到有相同 hash 值的文件。

（4）服务器则返回持有这个文件的用户信息。

（5）得到这个信息后，eMule 客户端就可以直接和拥有那个文件的用户沟通，看看是不是可以从他那里下载所需的文件。

（6）拥有那个文件（eMule 客户端发出搜索的文件）的用户可能不止一个，文件也可以是以片断的形式存在。

（7）在查找到下载源（其他客户端）后，客户端之间进行通信，交互文件内容，在下载文件的同时，也向其他需要此文件的客户端上传。

（8）下载过程是在客户端和客户端之间通过点对点（P2P）进行直接对话。期间没有数据流通过服务器。

注意：eMule 的工作过程中，它最棒的部分就在于，当你在下载文件的时候，不是只在一个用户那里下载文件，而是同时从许多个用户那里下载文件。如果另一个用户仅仅只有你要的文件的一个小小片断，他也会自动地把这个片断分享给大家，而你就可以从这个用户的机器上下载这个片断。当然你也是一样，只要你得到了一个文件片断，系统就会把这个片断共享给大家。

在 eMule 的工作过程中，需要处理两个重要的步骤，一个是搜索，另一个就是下载，下面就分别讲一下，eMule 是如何搜索及如何下载的。

2．eMule是如何搜索的

每一个客户端连接到一个服务器作为他的主服务器。在连接时，由客户端告诉主服务器他共享了那些文件，以及 IP 地址等其他信息。

所以每一个服务器会记录所有登录到他服务器上的以上信息。在本服务器搜索时，它会通过匹配记录的已知信息把查找结果反馈给搜索的客户端列表。当使用扩展搜索（extend search）时，你的搜索请求和应答结果通过发送限制带宽的 UDP 包，连接到客户端本身的服务器列表（server.met）对应的某一个 IP 地址的服务器上。

3．eMule是如何下载的

当客户端选择了一个文件下载时，它首先收集一个拥有该文档的客户端的列表。它会先查询主服务器的所有登录用户看他们是否拥有该文件，然后再连接和查询其他服务器的登录用户所拥有该文件的客户端列表。

一旦它找到拥有该文件的其他客户端，将请求每个客户端发送这个文件的不同片。直

至最后，要下载的文件由各个来源不同的分片组装成一个完整的、可用的文件。

在 eMule 下载的过程中，用户执行暂停/复位操作时，eMule 客户端并不丢弃已经获取并选择的下载列表，它所暂停的仅仅是下载客户端与提供文件源的客户端之间的 TCP 连接，执行复位操作后，恢复 TCP 连接就可以接着原有的下载过程继续下载。

用户在执行暂停/复位操作时，可以不登录服务器，也不需要主服务器进行任何干预和操作，只需通过客户端向服务器端发送 22 个字节复位通知后即可，占用的仅仅是 22 个字节的网络流量。在通过 eMule 进行文件下载的时候，它并未占用主服务什么资源，只是在控制下载的过程中，与主服务器发生若干字节的交互。因而说，eMule 系统中，资源的交互只发生在客户端与客户端之间，服务器不保存任何文件，也不参与文件的下载。

7.3　eMule 协议分析

eMule 是流行的文件共享程序，发源于 eDonkey 协议。eMule 协议描述了 eMule 的网络行为解释了理解该协议所需的基本术语。eMule 协议规范是进行 eMule 相关开发的核心指导手册。本节就重点讲述一下 eMule 的网络协议规范的部分核心内容。读者想要知道此协议完整规范的原文，建议参考 eMule 官方发布的英文版协议原文。

注意：eMule 协议规范原文可从 eMule 官方网站获取，在一些 eMule 社区和论坛上有非官方的翻译版，有兴趣的读者可自行参阅。

7.3.1　eMule 协议概述

eMule 网络是由上百个 eMule 服务器和几百万个 eMule 客户端组成。客户端必须连接到一个服务器来取得网络服务，只要该客户端在系统中，服务器连接保持打开状态。

eMule 服务器主要执行集聚索引服务（类似于 Napster 的中央索引服务器），相互间不联系。eMule 客户端都预先配置了一个服务器列表和当地文件系统的共享文件列表。

客户端用单独的 TCP 连接到一个 eMule 服务器登录到网络中，获得想得到的文件信息和其他的客户端信息。获取这些信息后，eMule 客户端用几百个 TCP 连接到其他客户端进行上传和下载文件。

每个 eMule 客户端对它的每个共享文件都维护着一个上传队列。要下载的客户端先加入到队列的底部，然后逐渐前进直到到达队列的顶部并开始下载它的文件。一个客户端可以从几个不同的 eMule 客户端中下载同一个文件的不同文件块。

注意：客户端和服务器的交流都是基于 TCP 的。服务器使用了一个内部数据库，用来存储关于客户端和文件的信息。一个 eMule 服务器不存储任何文件，它为关于文件位置的存储信息作集聚索引。服务器还有另一个受争议的功能，就是连接由于通过防火墙连接而无法接收到连接的两个客户端。这个连接功能增加了服务器的负载。相对于服务器和其他客户端，eMule 使用 UDP 来增强客户端的能力。客户端发送和接收 UDP 信息的能力在日常使用中不是强制使用的，当有防火墙阻止它收发 UDP 信息时也能无瑕疵地运行。

客户端也可以上传它还没有完成的文件分块。总地来说，eMule 协议扩展了 eDonkey 的能力，允许客户端之间交换关于服务器、其他客户端和文件的信息。

7.3.2　eMule 协议的核心内容

eMule 协议规范中，有两个最核心的内容，一是 eMule 客户端到 eMule 服务器之间的连接规范，另一个就是多个 eMule 客户端之间的连接规范。

1．客户端到服务器的连接

eMule 客户端启动后会发一个 TCP 连接到一个 eMule 服务器。服务器在收到这个连接请求后会提供一个客户 ID 给客户端，在整个客户端←→服务器连接的生命周期里，它是有效的。在连接建立之后，客户端分别将它所拥有的共享文件列表发给 eMule 服务器。服务器就会把这个列表存储到它的内部数据库中。eMule 服务器中的这个数据库，通常包含了成百上千个有效的文件和活动的客户端

注意： 客户端 ID 有高 ID 和低 ID 之分，如果客户端有一个高 ID，它会从所有的服务器中接收到相同的 ID，直到它的 IP 地址改变。关于客户端 ID 在后文会有详细的讲解。

eMule 客户端还会向 eMule 服务器发送它的下载列表，这个列表包含着它想下载的文件。这个连接也是基于 TCP 建立的，连接建立成功之后，eMule 服务器给客户端发送拥有它想下载文件的其他客户端列表（这些客户端称作“源”）。从这点起，eMule 客户端就开始与其他客户端建立连接了，如图 7.13 所示。

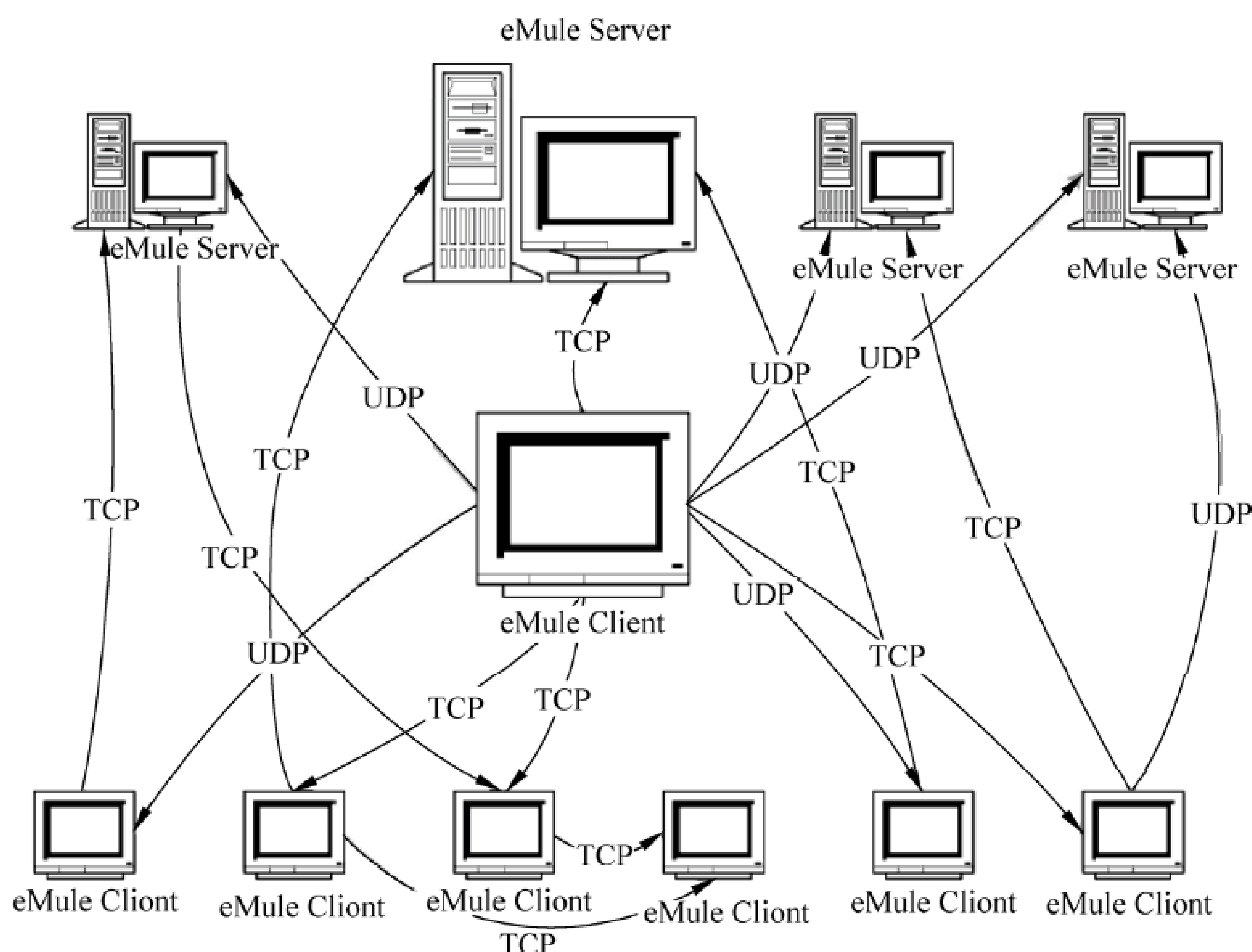


图 7.13　eMule 网络的连接关系

注意：在整个客户端会话期间，客户/服务 TCP 连接一直保持连接状态。初次握手后主要是用户活动激发事务，有时，客户端发送文件搜索需求，由搜索结果回应，一个搜索事务一般在对源中指定文件查询之后，用源（IP 和端口）列表来回答这个查询，查询者可以从这列表中下载文件。

客户端和它没有连接的服务器的交流是用 UDP。UDP 信息的目的是增强文件搜索，也就是通过 UDP 信息使 eMule 客户端可以向一个下止 eMule 服务器发起搜索，这样可以增强源搜索的能力，最后保持连接状态（确保客户端服务器列表中的 eMule 服务器有效）。

2．客户端到客户端的连接

eMule 协议的另一个重要内容就是规范了 eMule 客户端到客户端之间的连接，一个 eMule 客户端连接到另一个 eMule 客户端（源）是为了下载文件。一个文件分成很多部分。客户端可以从多个（不同的）客户端中下载同一个文件的不同的文件碎片，最后获得一个完整的文件。

当两个客户端连接时，它们交换容量信息，然后协商一个下载（或者上传）的开始。每个客户端有一个下载列表，记住一列等待下载文件的客户端。当 eMule 客户端下载队列空的时候，一个下载请求很可能会导致一个下载开始。当下载队列不是空的时候，就会将这个请求的客户端加入到队列中。

当下载的客户端到达下载队列的头部时，上传的客户端初始化一个连接来给它发送需要的文件块。eMule 客户端可以在几个其他客户端的等待队列中，都注册下载相同文件的块。当一个等待的客户端完成了（从它们中的一个）下载文件块时，它不会通知其他客户端在其队列中删除它，当它到达它们的队列头时只是简单地拒绝它们的上传意图。

eMule 用一个信用系统来鼓励上传，为了防止假冒用 RSA 公钥密码系统来保护信用系统。客户端连接可能用一套 eDonkey 协议没有定义的信息，该信息称作扩展协议。扩展协议用来实施信用系统，一般信息的交换（像服务器和源列表的更新），通过收发压缩的文件块来改善性能。

7.3.3　eMule 协议中几个重要概念理解及程序实现

在 eMule 协议中有几个很重要的概念，初学者很容易模糊和混淆，现在把这个概念单独提出来，并对其中一些主要概念中涉及的算法思想用程序来进行实现，这对理解整个 eMule 协议有重要意义。

1．客户ID

客户 ID 是服务器在它们连接握手时提供的一个 4 字节标识符。客户 ID 只在客户-服务器 TCP 连接的生命期中有效，在客户端有一个高 ID 的情况下，如果它的 IP 地址发生改变，所有的服务器都会重新分配它同样的 ID 值。

在 eMule 协议中，客户 ID 的分配具有重要的作用，客户端 ID 有低 ID（LowID）和高 ID（HighID）之分。当一个客户端不能接收一个输入连接时，eMule 服务器将分配给客户端一个低 ID。拥有一个低 ID 会限制客户端对 eMule 网络的使用，并且在实际应用中可能

导致服务器拒绝一个客户端连接（下文会有实例讲解）。

高 ID 的计算是以客户端 IP 地址为基础的，如果客户端自由地连接到其本机上的 eMule 的 TCP 端口（默认端口号是 4662），那么就会被分配给一个高 ID。有高 ID 的客户端没限制使用 eMule 网络。

在上文中已经讲过 ID 的计算方法，这里再强调一遍。高 ID 用下面的方法计算：假设主机 IP 是 X.Y.Z.W，ID 就是 X+2^8*Y+2^16*Z+2^24*W，如果这个值小于 16777216（0x1000000），那么它就是低 ID，反之就是高 ID。

注意：关于这个分界值 16777216 是怎么计算得来的，还没有找到什么文献来证明，只是在不同的服务器中会得到不同的低 ID。

当服务器无法打开一个 TCP 连接到客户端的 eMule 端口时，会分配一个低 ID 给该客户端。这主要发生在机器上装有防火墙的客户端，阻止了输入连接。当出现下面情况时，客户端也会接收到一个低 ID。

- 当客户端通过 NAT 或代理服务器连接时。
- 当服务器繁忙（导致服务器重连接计时器超时）时。

低 ID 客户端没有其他客户端可以连接到的公网 IP，这样所有的交流必须通过 eMule 服务器完成。这增加了服务器计算能力的负担，并且导致服务器勉强接收低 ID 客户端。这也意味着低 ID 客户端不能连接到不在同一个服务器上的其他低 ID 客户端，因为 eMule 不支持在服务器间的管道连接。

为了支持低 ID 客户端，引入了回调机制。使用这种机制，高 ID 客户端请求（通过 eMule 服务器）低 ID 客户端连接它来交换文件。

下面从程序的角度来说一下，如何用程序来实现客户 ID 的算法，也就是怎样通过一个 IP 值返回给用户一个 ID 号。

以下示例代码是对 eMule 协议中一些重要概念的具体实现，本文列举的方法只是一些主要方法的简单说明。在本书的随书光盘中有详细的、可运行的源代码。

【源代码示例\ch7\ch7_code\eMuleProtoco\ClientID.java】

程序清单：ClientID.java，用 Java 编程的方法来实现客户 ID 的算法。

```
/*
 * 类名：ClientID.java，用 Java 程序实现 eMule 协议中 ClientID 的算法
*/
import java.nio.*;                          //导入相应的程序包
/
public class ClientID {
    //定义字节数组的数据结构，用来存储用户的 ID，用户 ID4 个字节长度的
    private byte[] data = new byte[4];
    /**
     * 构造方法
     * @param clientID，字节类型
     */
    public ClientID(byte[] clientID){  //接收一个字节数组作为参数的构造方法
        for(int i = 0;i<clientID.length;i++)    //遍历字节数组，将值写入到已定
                                                 义的数据结构中
            this.data[i] = clientID[i];         //赋值
    }
```

```
    /**
     * 构造方法
     * @param clientID，字符串类型
     */
    public ClientID(String clientID){      //接收一个字符串作为参数的构造方法
        data = (stringIPToArray(clientID));//将参数字符串转换为字节数组
    }
    /**
     * Method name:getClientID
     * TODO : 取得 ClientID 的值
     */
    public byte[] getClientID() {  //从定义的数据结构中取出 ClientID 的数据值
        return data.clone();       //返回定义的数据结构中的复制值即可
    }
    /**
     * Method name:isHighID
     * TODO : 判断此 ID 值是否是高 ID
     */
    public boolean isHighID() {    //判断此 ID 是否高 ID
        return !(data[3]==0);   //高 ID 的判定算法，只需根据第三个字节的值来判定
    }
    public String getAsString() {  //将 ClientID 值以字符串的形式返回
        return byteToInt(data[0])+"."+ byteToInt(data[1])+"."+ byteToInt
        (data[2])+"."+ byteToInt(data[3]);
    }
```

以下是对 eMule 协议中对计算 ClientID 方法的具体实现，根据客户 ID 的计算规则，如果已知客户 ID 的 IP 地址为 x.y.z.w，那么 ID 值是 X+2^8*Y+2^16*Z+2^24*W，下面就是这个算法的具体实现。

```
/**
 * eMule 协议中，对 ClientID 的计算算法
 */
public int hashCode() {              //根据 ClientID 的计算方法来计算 ID 值
    long num = data[0];              //取出第 1 个 IP 段的值
    num+=Math.pow(2, 8)*data[1];     //计算第 2 个 IP 段的值
    num+=Math.pow(2, 16)*data[2];    //计算第 3 个 IP 段的值
    num+=Math.pow(2, 24)*data[3];    //计算第 4 个 IP 段的值
    return longToInt(num);           //将各个 IP 段的值求和后返回
}
public boolean equals(Object object){//重写 equals()方法，用来判定两个 ID 值的关系
    if (object==null) return false; //如果传入的参数为 null，直接返回 false
    if (!(object instanceof ClientID)) return false;//如果不相等也返回 false
    return this.hashCode()==object.hashCode(); //返回此对象的 hashCode()
}
```

以下是不同数据之间的转换方法，因为在计算客户 ID 的时候，数据结果与数据处理过程之间，数据类型及格式都是不相同的，所以需要在不同的数据之间进行相互转换。以下就是不同数据类型之间转换的具体实现。

```
    /**
     * Method name:byteToInt
     * TODO : 数据类型转换，将 Byte 转换成 Int 类型
     */
```

```
   public int byteToInt(byte bvalue) {
     //分配一个 ByteBuffer 空间，并初始化大小为 4 个字节，在 Java 中 int 类型数据为 4
     字节长
       ByteBuffer data = ByteBuffer.allocate(4);
       //根据 eMule 协议规定，使用 LITTLE_ENDIAN 的编码字节序
       data.order(ByteOrder.LITTLE_ENDIAN);
       //将数据置入 ByteBuffer 空间
       data.put(bvalue);
       //返回一个 int 类型的值
       return data.getInt(0);
   }
   /**
    * Method name:longToInt
    * TODO : 数据类型转换，将 Long 转换成 Int 类型
    */
  public int longToInt(long value){
      //分配一个 ByteBuffer 空间，初始化大小为 8 个字节，在 Java 中 long 类型数据为 8
        字节长
       ByteBuffer data = ByteBuffer.allocate(8);
     //同样将其转换为 LITTLE_ENDIAN 的字节序编码
       data.order(ByteOrder.LITTLE_ENDIAN);
     //调用 putLong()方法，将一个 Long 型数据置入 ByteBuffer 空间
       data.putLong(value);
       //返回一个 int 类型值
       return data.getInt(0);
   }
   /**
    * Method name:intToLong
    * TODO : 数据类型转换，将 int 转换成 Long 类型
    */
  public long intToLong(int value){
    //分配一个 ByteBuffer 空间，初始化大小为 8 个字节，在 Java 中 long 类型数据为 8 字节长
       ByteBuffer data = ByteBuffer.allocate(8);
       //转换编码字节序
       data.order(ByteOrder.LITTLE_ENDIAN);
       //将值置入 ByteBuffer
       data.putInt(value);
       //返回一个 Long 型的转换结果数据
       return data.getLong(0);
   }
   /**
    * 将 IPv4 地址格式转换成字节数组，接收一个字符串的 IP 地址作为参数
    * A.B.C.D = [A,B,C,D]
    */
   public byte[] stringIPToArray(String IPAddress){
       int j = 0;
       byte[] data = new byte[4]; //初始化一个 4 字节大小的 Byte 数组空间
       for(int i=0;i<3;i++){       //处理点分十进制格式的字符串表示的 IP 地址
           String p="";            //通过 IP 地址中的"."符来进行分段处理
           while (IPAddress.charAt(j)!='.') p=p+IPAddress.charAt(j++);
           j++;
           //对每一个地址段进行处理，以"."为标记，只处理前 3 个段
           data[i]=(byte)Short.parseShort(p);
       }
       String p="";
        //处理 IP 地址中的最后一段
       for(int i=j;i<IPAddress.length();i++) p=p+IPAddress.charAt(i);
```

```
        //将处理后的值置入字节数组中的最后一个字节
        data[3]=((byte)Short.parseShort(p));
        //将转换处理后的值返回
        return (data);
    }
}
```

2. 用户ID（用户哈希）

eMule 支持信用系统来鼓励用户共享文件。用户上传越多的文件给其他客户端，则接收的信用越多，在它们的等待队列中前进得越快。

用户 ID（也叫用户哈希）是 128 位（16 字节）、连接随机数字创建的 GUID，第 6 和第 15 字节不是随机产生的，它们的值分别是 14 和 111。

在客户 ID 中已经讲过，在整个客户端和指定的服务器会话中，客户 ID 是有效的，然而用户 ID 是唯一的并且跨越会话时用来识别客户端（用户 ID 识别工作站）。用户 ID 在信用系统中扮演重要角色，这为“黑客”假冒其他用户来获得他们信用赋予的优先权提供了动机。eMule 提供加密方案设计来阻止欺骗和冒名顶替。这个实施是简单的应答交换，依靠 RSA 公有/私有钥匙加密。

在实际的应用中，要用程序来实现用户 Hash 也很简单，只需根据算法规则进行相应的计算即可，在随书光盘中提供了关于用户 Hash 算法实现的详细源代码。

【源代码示例\ch7\ch7_code\eMuleProtoco\UserHash.java】

以下是 UserHash.java 类中主要方法的说明，此方法主要用 Java 编程的来实现用户哈希的算法。

```
/**
 * 类名，UserHash.java,实现 eMule 协议中对用户 ID（用户哈希）的算法。
*/
import java.nio.*;                          //引入 Java 中与 IO 操作有关的包
import java.util.*;                             //引入 Java 的工具包
public class UserHash {                         //定义一个 UserHash 类
    private byte[] userHash = new byte[16];//初始化一个 16 字节大小的字节数组
    public UserHash() { }                       //构造方法
    public UserHash(byte[] hash){               //需要传入一个字节数组的构造方法
        this.userHash=hash;
    }
    public UserHash(String hash) {              //需要传入一个字符串 Hash 的构造方法
        loadFromString(hash);
    }
    /**
     * Method name:genNewUserHash
     * TODO : 根据 eMule 协议规范的规定，计算用户的 Hash 值
     */
    public static UserHash genNewUserHash() {
        byte[] hash = new byte[16];             //定义一个 16 字节的 Hash 结构
        new Random().nextBytes( hash );         //取随机数的过程。
        //第 6 和第 15 字节的值是规定好的，必须是 14 和 111
        hash[5] = 14;                           //设定其中第 5 个字节的值为 14
        hash[14] = 111;                         //设定其中第 14 个字节的值为 111
        return new UserHash(hash);              //返回求得的用户 Hash 值
    }
```

🔔注意：以上的方法只是抽取了 UserHash.java 类中的主要方法进行说明，读者在学习这个知识的时候，要结合 eMule 协议的算法规则和程序源码，实际地执行、验证一下，这样会理解的更深刻。

3．文件ID

文件 ID 用来唯一地标识网络中的文件和文件损坏侦测和修复。文件是用由客户端和基于文件内容计算出来的 128 位 GUID 哈希来标识的。GUID 是应用 MD4 算法到文件数据中计算而来。当计算文件 ID 时，文件被分成每段 9.28MB 长的部分。每部分单独计算出一个 GUID，然后所有的哈希组合成一个唯一的文件 ID。

当下载的客户端完成一个文件部分下载时，它计算这部分哈希，然后和发送过来的这部分哈希对比。如果发现这部分损坏了，客户端将尝试通过逐渐替换这部分中的位（每个 180kb）来修复损坏部分，直到哈希计算完成。

🔔注意：eMule 不依靠文件名来唯一标识和编码文件，而通过哈希文件内容计算出 GUID 标识文件。有两种类型文件 ID，一种主要用来产生唯一的文件 ID，另一种是用来损坏侦测和修复。

根哈希用 SHA1 算法来为每部分计算根哈希，基于每块 180kb 大小。它提供了更高等级的可靠性和可修复性，更多信息可在 eMule 官方网站得到。关于文件 ID 的计算方法，本文也提供了相应的程序实现。

【源代码示例\ch7\ch7_code\eMuleProtoco\FileHash.java】

以下是 FileHash.java 类中主要方法的说明，本方法依据 eMule 协议中关于文件 ID 的算法规则，以实现计算文件哈希值的目的。

```
/*
 * 类名：FileHash.java，用于实现 eMule 协议中，对文件哈希的算法，以计算出文件的 Hash 值
*/
import java.nio.*;
public class FileHash {
    private byte[] fileHash;        //字义文件 Hash 的结构，用字节数组来存储
    public FileHash(byte[] fileHash){//文件 Hsah 的构造方法，传入字节数组作为参数
        this.fileHash = fileHash;
    }
    public FileHash(String inputString){
                                   //文件 Hsah 的构造方法，传入字符串作为参数
        fileHash = new byte[16];    //初始化文件 Hash 空间的大小，为 16 字节
        for(int i = 0; i < fileHash.length; i++) //对每一个字节进行处理
        fileHash[i] = hexToByte(inputString.charAt(i*2)+""+inputString.
        charAt(i*2+1));
    }
    //以下几个方法主要用于处理不同数据类型之间的转换
    /** 将 0xAA 形式的十六进制值转换成 Int 类型的值 */
    public int hexToInt(String value){
        return Integer.parseInt(value.charAt(0)+""+value.charAt(1), 16);
    }
}
```

注意：以上只是对主要方法的实现思想进行了简单的说明，这些方法的实现需要依赖很多其他的方法，完整的实现过程，请读者参考源代码。

4．eMule协议扩展

尽管 eMule 完全兼容 eDonkey，但它还是实行了几种扩展，允许 eMule 两个客户端为用户提供另外的功能。扩展只要集中在客户端与客户端的交流，特别是在安全和 UDP 使用领域上。关于 eMule 的协议扩展，不是本文讲解的重点，所以不再赘述。

5．eMule的消息编码格式（message encoding）

在 eMule 协议规范中，对 TCP/UDP 有效负载中消息编码进行了详细定义，在这里只对一般消息编码要点（General message encoding issues）进行说明。

注意：在 eMule 协议规范中，对不同状态、不同对象之间传递的消息格式都有严格的定义，本章不再对每个具体的消息格式进行说明。在进行 eMule 系统开发时，这些消息格式是必需掌握的。

（1）eMule 消息的字节序：所有消息都是用 little-endian 编码，而不是 big-endian，big-endian 是约定的网络字节序。这个很容易解释，事实上客户端/服务器都是基于微软窗口的应用程式，运行在 Intel 处理器上，应用这样的字节序没有什么难以理解的。

注意：big endian 和 little endian 是 CPU 处理多字节数的不同方式。例如“汉”字的 Unicode 编码是 6C49，那么写到文件里时，究竟是将 6C 写在前面，还是将 49 写在前面？如果将 6C 写在前面，就是 big endian。如果将 49 写在前面，就是 little endian。

endian 这个词出自《格列佛游记》。小人国的内战就源于吃鸡蛋时是究竟从大头（Big-Endian）敲开还是从小头（Little-Endian）敲开，由此曾发生过六次叛乱，其中一个皇帝送了命，另一个丢了王位。我们一般将 endian 翻译成“字节序”，将 big endian 和 little endian 称作“大尾”和“小尾”。关于字节序、网络字节序的概念请参考本书第 6 章的相关内容。

（2）eMule 消息头的定义：在 eMule 系统中，所有的用于交互的消息都有一个 6 字节的头，消息头有下面的结构。

- 协议：一个字节，其中协议 ID 为 0xE3 表示的是此消息遵循 eDonkey 协议，而协议 ID 为 0xC5 的时候，表示的是 eMule 协议。
- 大小：4 个字节，指消息的大小，以字节为单位，不包含头。例如，如果消息不包含任何有效负载，则消息长度是 0。
- 类型：一个字节，类型指独一无二的消息 ID。

（3）eMule 消息标签：eMule 的消息标签类似于 TLV（类型，长度，值）结构，用来增加可选的数据到 eMule 消息中。在这种标签中，每个标签都拥有 4 个域，在消息中它们并不都是连续的，消息标签的结构如下。

- 类型：1 字节整型；

- 名称：可以是以下之一
  - 可变长度字符串；
  - 1 字节整型；
- 值：可以是以下之一
  - 4 字节整型；
  - 4 字节浮点数；
  - 可变长度字符串；
  - 专用的- 1 字节整型。

注意：当提及到协议消息里特定的标签时，只有标签类型是指定的，读者应该参考完整的 eMule 协议规范原文来确定协议消息的准确的结构。

专用的标签指定者带有整型值的标签叫做整型标签，类似的字符串标签和浮点数标签。字符串标签的类型值是 2，整型标签的类型值是 3 而浮点数标签的类型值是 4。

当标签被编码发送时，是按照上面的次序编码，例如，类型然后名称，最后是值。类型被编码成一个字节。名称被编码成 2 个字节长度值，可以是字符串名称和整型名称。

注意：标签给出的名称是没有特定协议意义的，只是易于在以后的协议消息描述中引用。

在 eMule 协议规范中，在活动用户数量的服务器配置中有两种限制，分别为软件和硬件。硬件限制远大于软件限制。当活动用户的数量达到软件限制时，服务器停止接收新的低 ID 客户连接。当用户数量达到硬件限制时，服务器满了，不再接收任何客户端连接。这主要是针对 eMule 服务器而言的，读者只需了解。

7.3.4　eMule 协议分析之——客户端与服务器之间的 TCP 通信

只要客户端一启动，就会与一个确切的 eMule 服务器建立 TCP 连接。eMule GUI 客户端为了操作，要求服务器建立连接，但客户端不能同时和几个服务器建立连接。所以，客户端与服务器之间的 TCP 通信，就是客户向其中之一的 eMule 服务器发送 TCP 消息进行信息交互的过程，这个过程的详细实现过程如下。

1．建立连接

当客户端和服务器建立连接时，可能同时尝试和几个服务器建立连接，直到获得一个服务器成功注册（login）的响应后就放弃所有其他的连接请求。建立连接可能有如下情况。

- 高 ID 连接：服务器给连接客户端分配高 ID；
- 低 ID 连接：服务器给连接客户端分配低 ID；
- 拒绝会话：服务器拒绝客户端。

当然也有个别情况下，服务器崩溃造成无法连接，或是超出软硬件的限制而拒绝连接。

2．连接启动信息交换

在建立成功的连接后，客户端和服务器交换几个设置消息。这些消息的目的是互相更新它对等点的状态。

客户端向服务器发送它的共享文件列表，然后服务器将列表存放在其内部数据库中。服务器发送它的状态和版本，然后发送它所知的 eMule 服务器列表和提供更多一些自我认定的细节。

最后客户端要求源（可以访问下载它下载列表中的文件的其他客户端）和服务器回应一系列的消息，客户端下载列表中的每个文件，直到下载所有的源列表到客户端。如图 7.14 所示为连接启动信息交换的过程。

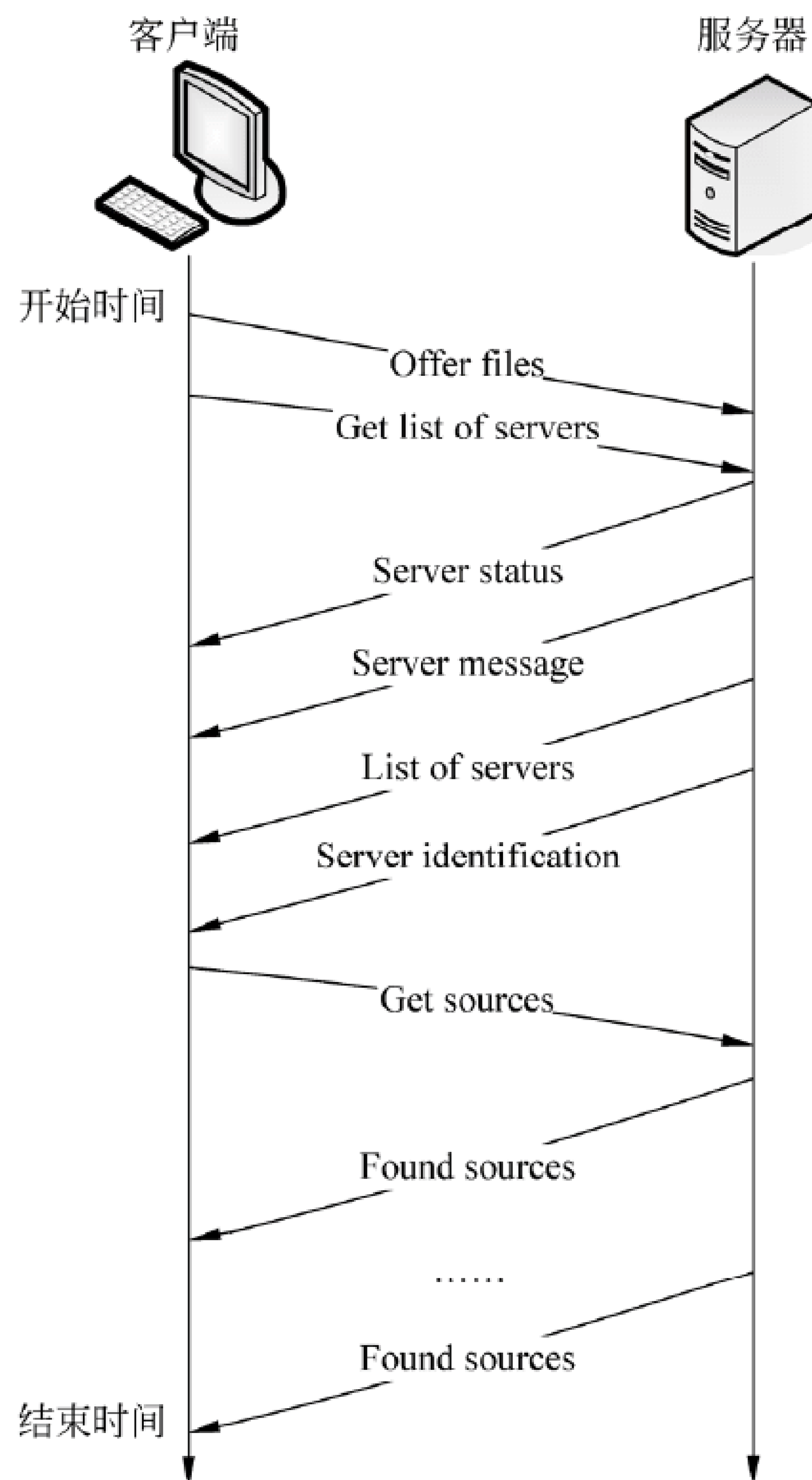


图 7.14　连接启动信息交换示意图

3．文件搜索

在整个客户端的会话中，客户端与服务器的 TCP 连接一直保持打开。在初始化握手后，大部分事务是由用户事件（user activity）发起，也就是说，文件搜索是由用户发起的，当一个搜索要求发送到服务器后，服务器用一个搜索结果回应。当有很多结果时，搜索结果消息就会被压缩。

一般在一个查找事务之后，会有一个针对某个特定文件的“源”的查询，服务器的应答消息是一个关于源（IP 和端口）的列表。在检验出“源”是新的之后，eMule 客户端开始尝试连接和把它们加入到它的源列表。根据这个列表，发送请求客户端就可以从中下载文件了。如图 7.15 描述了文件搜索时的信息交换过程。

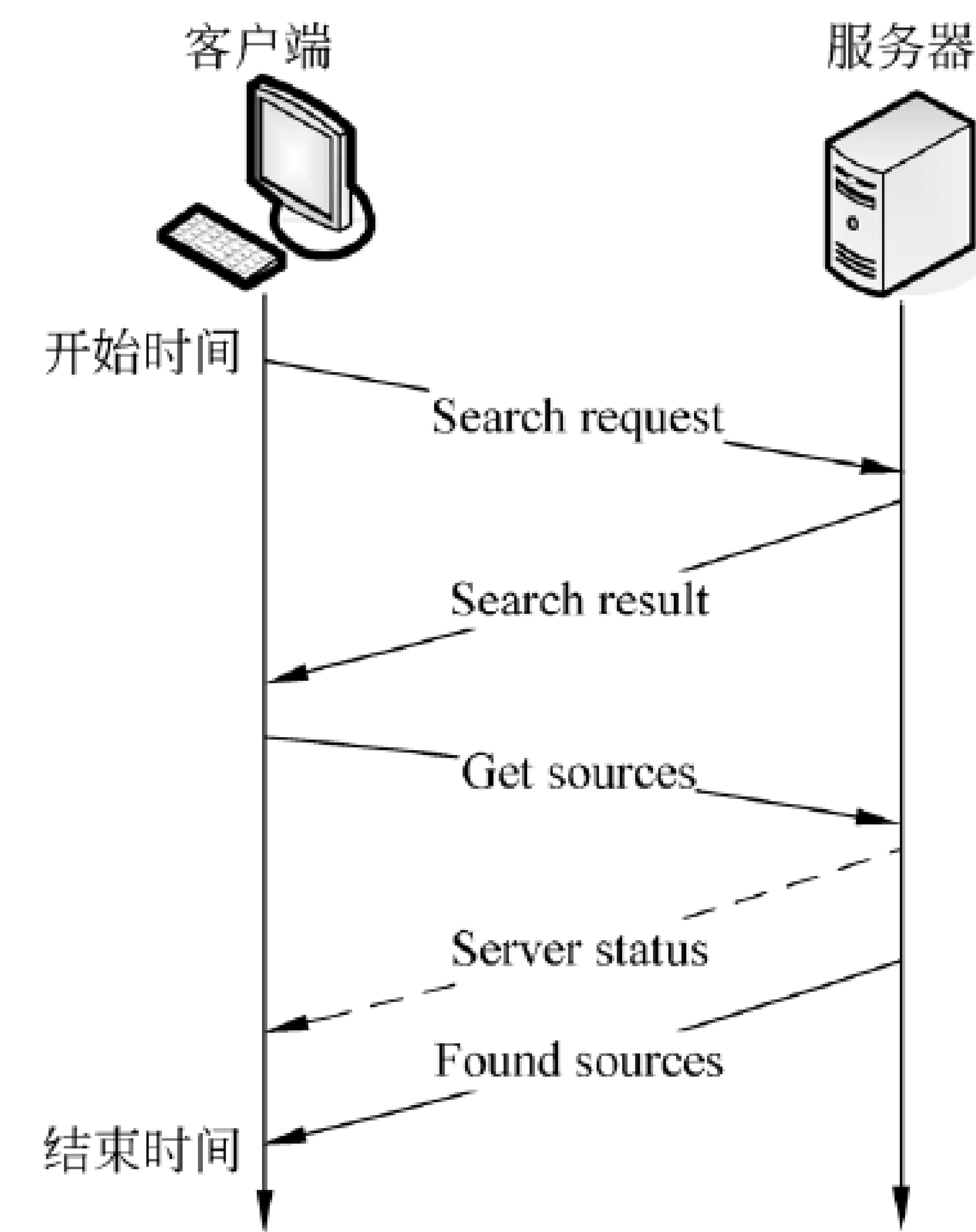


图 7.15　文件搜索过程信息交换示意图

4．回调机制

回调机制是设计来克服低 ID 客户端不能接收输入的连接的，低 ID 客户端限制使用 eMule 网络并且导致没有公有 IP 能和其他客户端相连，因此所有的通信必须通过 eMule 服务器来进行中转。

这意味着低 ID 客户端不能与不在同一个服务器中的低 ID 客户端连接，因为 eMule 不支持服务器之间的隧道请求。为了克服低 ID 客户端不能接受连接，从而不能共享其他客户端文件，设计了回调机制，这样客户端之间就能共享它们的文件。

回调机制的实现很简单，其消息交换示意图如图 7.16 所示。

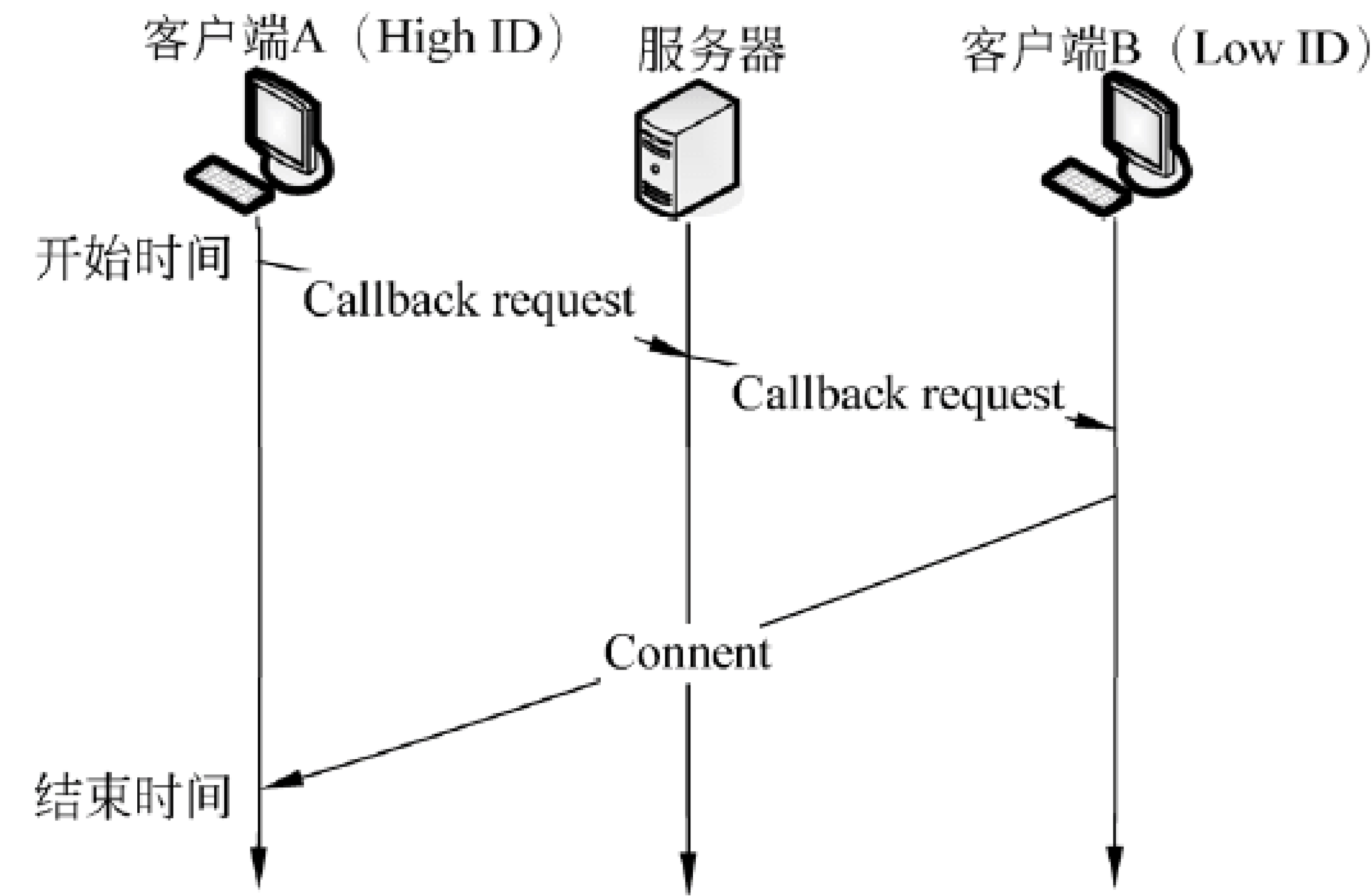


图 7.16　回调机制的信息交换示意图

假如客户端 A 和 B 都连接到同一个 eMule 服务器，A 需要的文件在 B 上，但 B 是低 ID 的，A 可以向服务器发送一个回调请求，请求服务器作为中间人，让 B 来主动的呼叫 A。服务器此时已经有一个与 B 打开的 TCP 连接，当有回调请求时，服务器会发送一个回调请求消息到 B，为 B 提供 A 的 IP 和端口。得到 A 的 IP 和端口以后，尽管 B 是低 ID 的用户，

但是 B 却可以主动发起到 A 的连接，并向 A 发送文件。这一过程并没有给服务器增加负担。很明显，只有高 ID 客户端可以要求低 ID 客户端回调（低 ID 客户端是没有接收输入连接能力的）。

7.3.5　eMule 协议分析之——客户端到服务器端 UDP 通信

eMule 客户端和服务器用不可靠的 UDP 服务来保持连接和增强搜索，UDP 消息可以增强文件查询和“源”查询的功能，同时提供保持在线（keep alive）功能（确保客户端的服务器列表中的服务器是有效的）。

eMule 客户端产生 UDP 包的数量（包不是字节），最多能达到 eMule 客户端发送所有包的 5%。这依赖于在客户端服务器列表中服务器的数量，以及客户端的下载列表中“源”的数量和用户执行搜索的数量。

UDP 包通过计时器触发，计时器每 100ms 过期，有一个单独的线程负责发送 UDP 输送结果，以每秒 10 个 UDP 的最大速率。

1．服务器保持在线（keep alive）和状态信息

通过 UDP 服务器的状态请求和 UDP 服务器描述请求，客户端周期性验证它服务器列表中的服务器状态。验证是通过发送 UDP 服务器状态请求和 UDP 服务器描述请求消息完成的。简单的保持在线配置是每 MIN 不超过 12 个包，任何情况下，包的最大速率是 0.2 个/s（或每 5s 发送一个包）。客户端在发送第一个服务器状态请求信息的同时监听服务器的状态，所以每隔 2 次才有一次服务器描述请求，描述如图 7.17 所示。

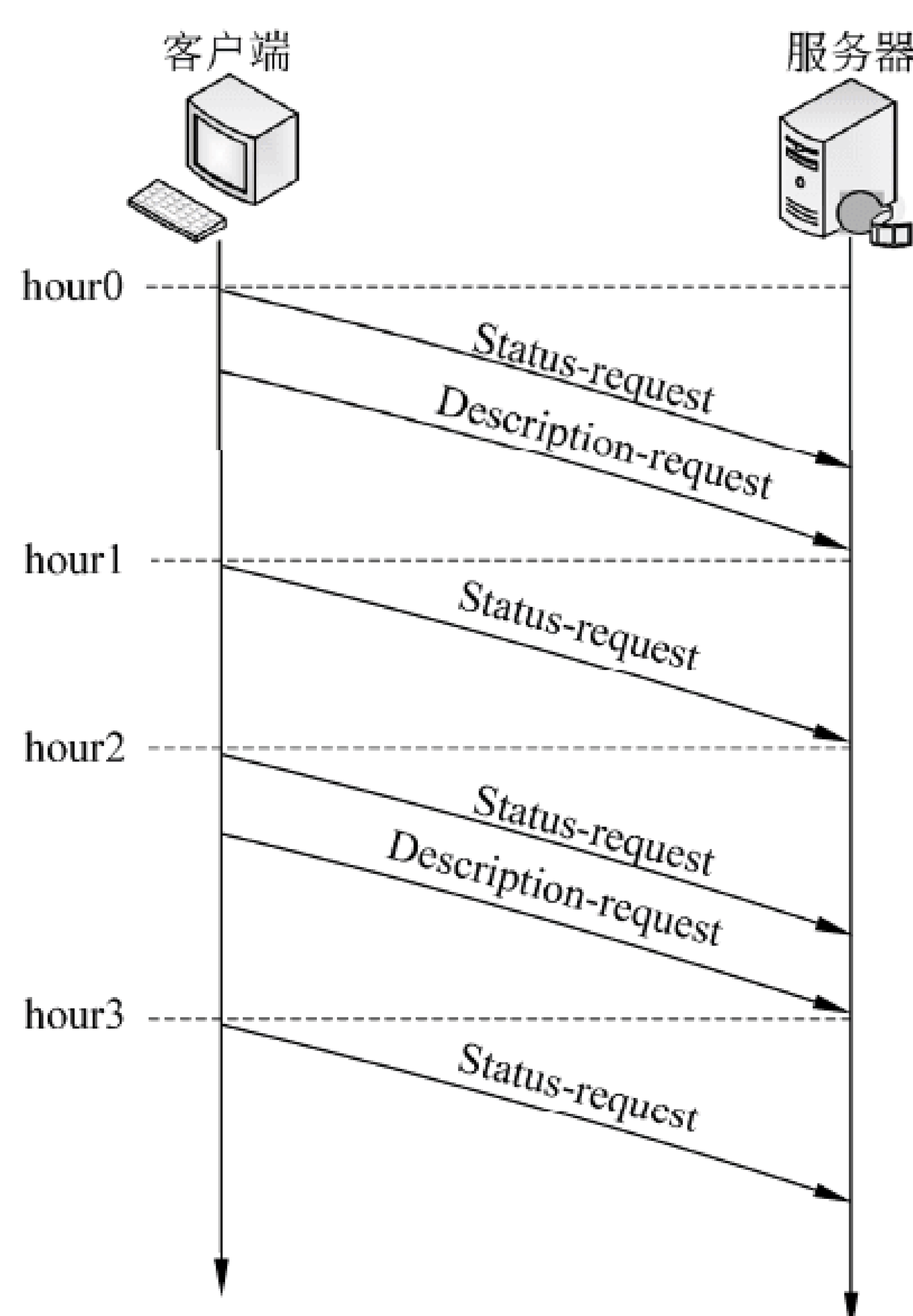


图 7.17　eMule 保持在线的 UDP 信息交换过程

2．增强文件搜索

eMule 客户端可以设置用 UDP 来增强它的文件搜索，使用 UDP，eMule 客户端可能会被重新配置，提高它的文件搜索。服务器回复的仅仅是它的搜索结果。UDP 搜索包会被发送到客户端服务器列表中的服务器中。

3．增强文件源搜索

当客户端要下载列表中某个文件的时候，如果该文件的来源少于一个可配置的限制(100)时，客户端就会周期性地发送 UDP 包给它服务器列表中的服务器来找更多的文件源。由客户端的 UDP 触发器每秒发送一个包，使得“源”能搜索到更多的部分。由于文件搜索依赖客户端获得文件的来源，相对 TCP 源搜索，UDP 源搜索比较弱。

注意：与 TCP 源搜索相反，UDP 源搜索在文件搜索中减弱，对于指定的文件，只是依靠客户端拥有的源数目。

7.3.6　eMule 协议分析之——客户端到客户端 TCP 通信

在 eMule 客户端注册到服务器和向服务器查询文件和“源”之后，为了下载文件，eMule 客户端需要联系其他客户端。为每对｛文件，客户端｝创建一个专用的 TCP 连接。当特定的周期内（默认 40s）没有任何 socket 活动或者对方已经关闭了这个连接，那么这个连接就会关闭。

为了提供合理的下载速率，直到可能提供给它（和所有其他下载的客户端）至少最小允许速率（当前的硬编码常量设置为 2.4k/s），eMule 才允许客户端开始下载文件。

1．握手初始化

握手初始化是一个对称过程，双方都发送相同的信息给对方，客户端交换双方的信息，这些消息包括它们互相辨认的标识、版本和接收信息的能力。两个 eMule 客户端之间的握手包括，eMule 客户端初始化握手的扩充内容 UDP 信息交换，安全辨认和源交换的能力。图 7.18 所示的就是两个 eMule 客户端之间的握手过程。

注意：在这种握手消息中，参与的有两种类型消息，即 Hello 消息和 eMule 信息消息（请参考 eMule 协议规范关于各个消息的说明）。第一种是 eDonkey 的一部分，兼容 eDonkey 客户端，第二种是 eMule 独有的扩展客户端协议的一部分。

2．安全的用户身份认证

安全用户认证是 eMule 扩展的一部分。eMule 中有一个信用系统，其安全可靠体系使用 RSA 公钥密码。信用系统执行这些扩展协议，是为了通用信息交换（如服务器和“源”列表更新），提高发送和接受压缩文件碎片能力。

（1）eMule 的身份认证：如果客户端支持安全认证，就会在初始化握手之后立即执行。安全认证的目的是防止用户冒名顶替。当实施安全认证时，执行以下步骤。

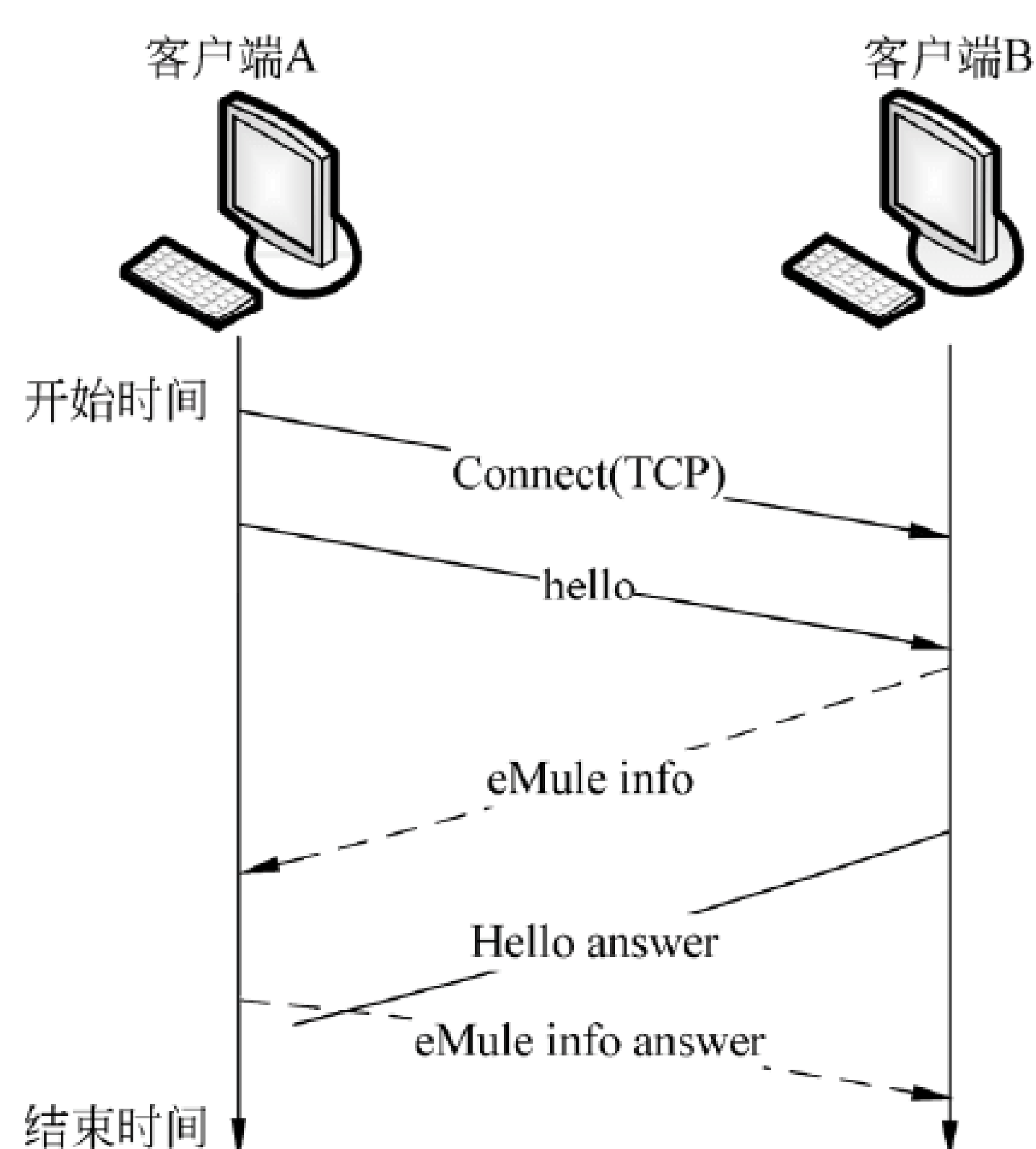


图 7.18　eMule 客户端之间的初始握手过程

- 在握手初始化时，B 客户端标明它支持安全辨认和希望用户安全辨认。
- 发送安全辨认信息的确认信息应标明 A 是否需要 B 公钥，包括 B 标记的 4 个字节。
- 若 A 标明它需要 B 公钥，B 就发送公钥给 A。
- B 发送一个签名（signature）信息，此消息是用发磅过来的询问和额外的双字节创建的，若 B 是低 ID 则此双字节附加 A 的 IP 地址；若 B 是高 ID 则附加 B 的 ID。

图 7.19 所示的就是安全的用户身份认证过程

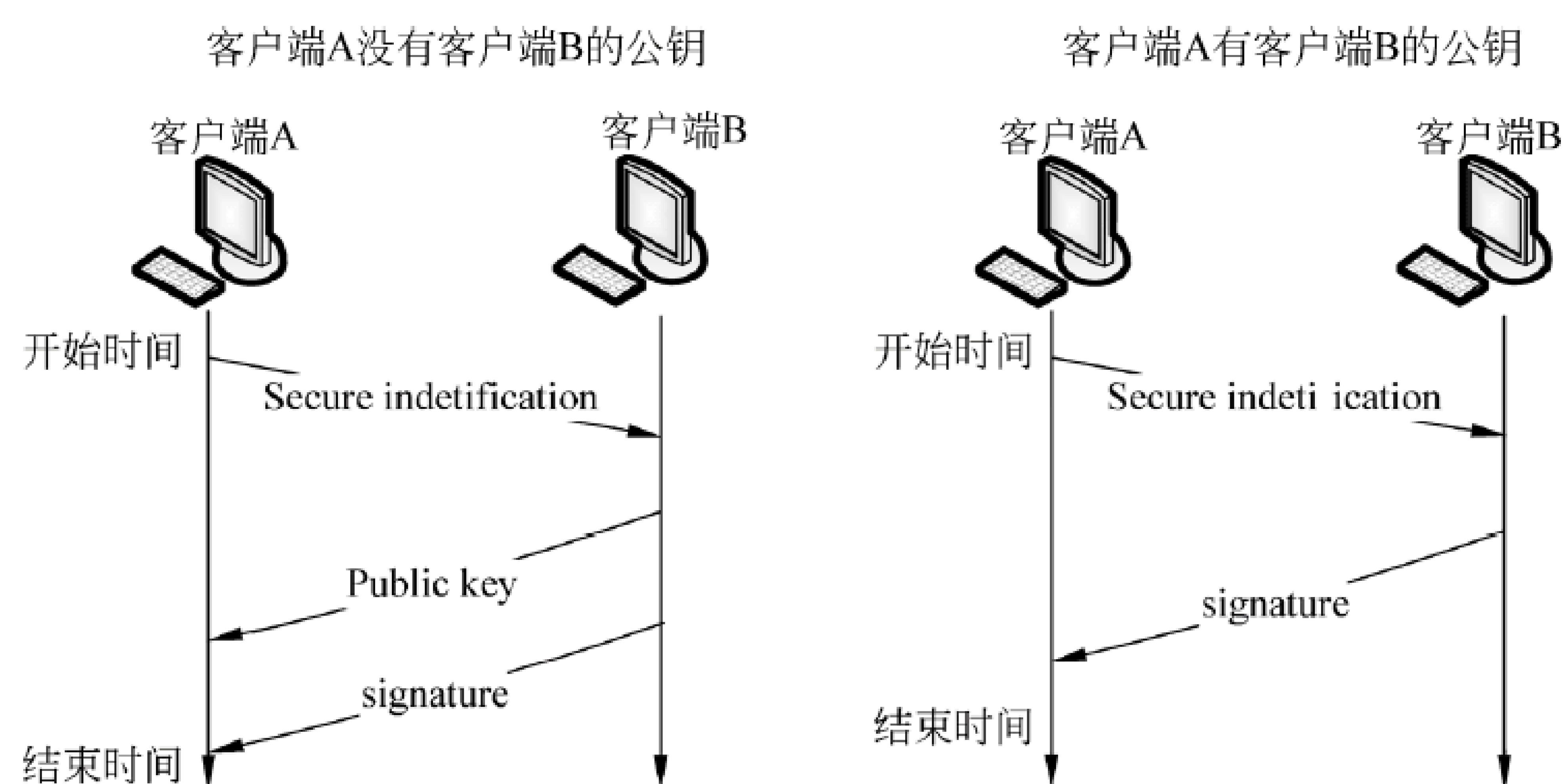


图 7.19　eMule 系统中安全的用户身份认证信息交换过程

（2）eMule 的信用系统：eMule 支持信用系统目的是为了鼓励用户共享文件，让用户上传更多的文件给其他客户端。

当客户端上传文件给它的对方，下载的客户端就根据数据传输的数量来更新它的信用。这个信用的体系不是全局的，只是确保下载客户端本地传输信用。

注意： 信用体系的非全局性指的是，传输的信用被下载的客户端局部保存，只有当上传的客户端（获得信用的那个）要求从这个特定的客户端下载时，信用才会被考虑。

（3）客户端信用的计算方法：信用的计算方法有两个表达式，信用的取值是这两个表达式计算结果的最小值。

假设用户在使用 eMule 下载时，其上传总量为 Upload_total，下载总量为 Download_total，信用值为 Credit_value，Min（a，b）为取最小值函数，即结果为 a，b 中的最小值。信用计算的两个方法分别如下。

- ❑ Method1：Upload_total×2/ Download_total，（当下载的总量是零时，这个表达式估值是 10）。
- ❑ Method2：$\sqrt{\text{Download_total+2}}$（当上传的总量小于 1MB，这个表达式估值是 1）。
- ❑ 则信用值 Credit_value=Min（method1，method2），其中（Credit_value<=10 and Credit_value>=1）

注意：上传/下载数量是以 M 为单位计算。任何情况下，信用不会高过 10 或者低于 1。

3．文件请求

每对｛客户端，文件｝都会创建一个独立的连接。在连接建立之后，客户端立即发送几个关于它希望下载的文件请求消息。一个典型、成功的请求消息过程如图 7.20 所示。

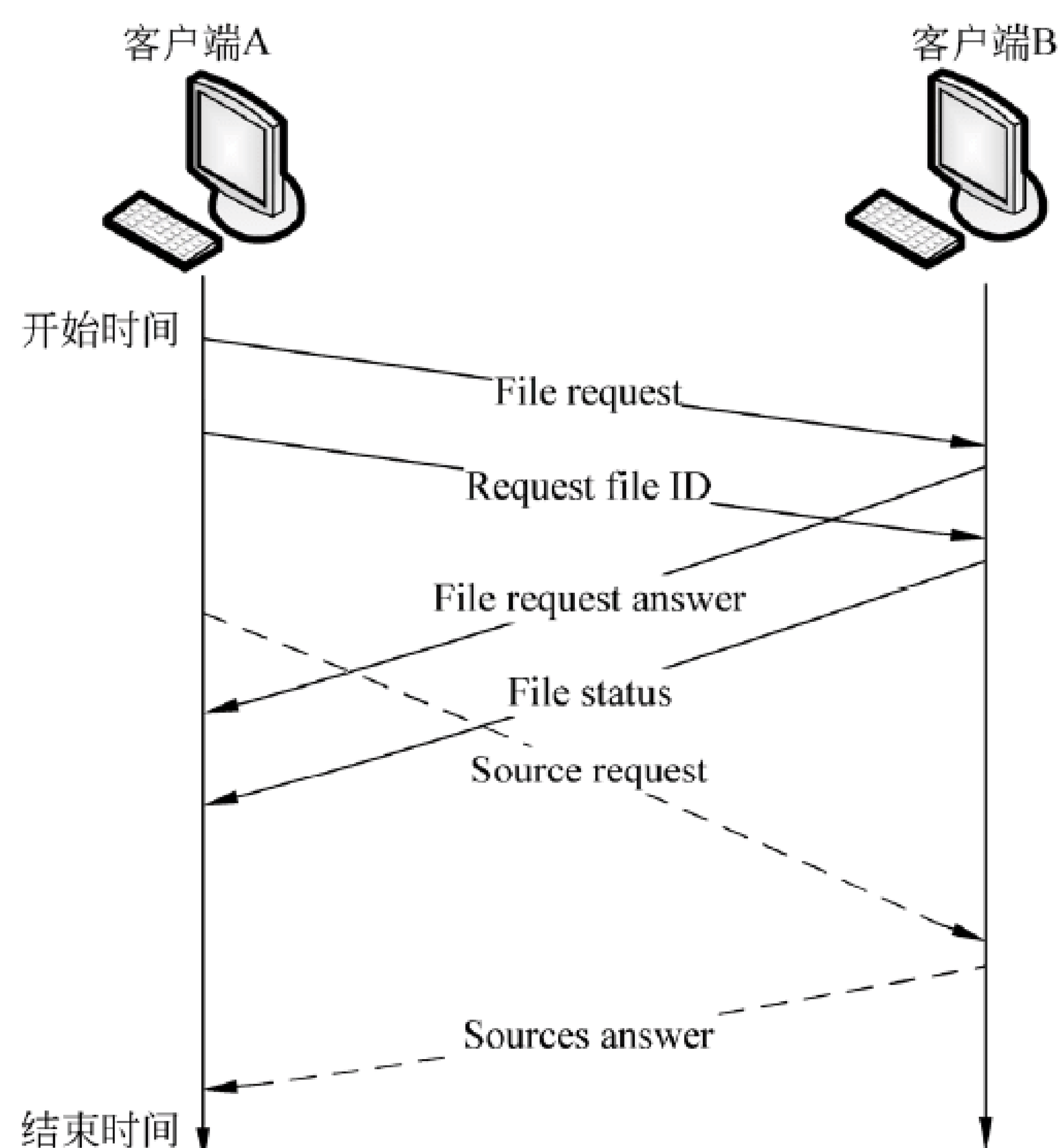


图 7.20　文件请求过程的信息交换

在图 7.20 所示的整个文件请求的过程中，需要完成基本信息交换、处理文件没有找到的情况、加入上传队列、上传队列管理等几个步骤，下面分别对这几个步骤进行说明。

（1）基本信息交换

在 eMule 系统中，两个对等的客户端 ClientA、ClientB，要完成基本信息的交换，有如下几个执行过程。

- ❑ A 发送文件请求消息和请求文件 ID 消息。

- ❑ B 分别对应发送文件请求回答消息和文件状态消息。

其中，文件请求回答消息来回应文件请求信息，而用文件状态消息来回应请求文件 ID 的消息。

在这个序列中，扩展协议增加了两个消息，“源”请求消息和“源”应答消息。假定 B 是当前正在下载的文件，eMule 就会用这个扩展来传递 B 的“源”给 A。

详细阐述就是，在 B 能够发送文件部分给其他客户端之前，B 没有必要完成文件下载，B 能发送给 A 任何部分，甚至它已下载的文件仅仅只是一个小小的文件碎片。若 A 向 B 请求一个文件，但 B 的共享文件列表中没有该文件，则 B 跳过文件请求回答信息，在请求文件 ID 消息收到后立即发送文件找不到的消息，同时关闭连接。

（2）没有找到文件的情况

当 A 向 B 请求一个文件时，但是 B 的共享文件列表中没有这个文件，B 会忽略这个文件的请求回应消息。在请求文件 ID 消息之后，立即发送一个没有文件的消息给 A，告诉 A 文件没有找到（File not found）。然后 A 与 B 之间的连接关闭。这个请求过程如图 7.21 所示。

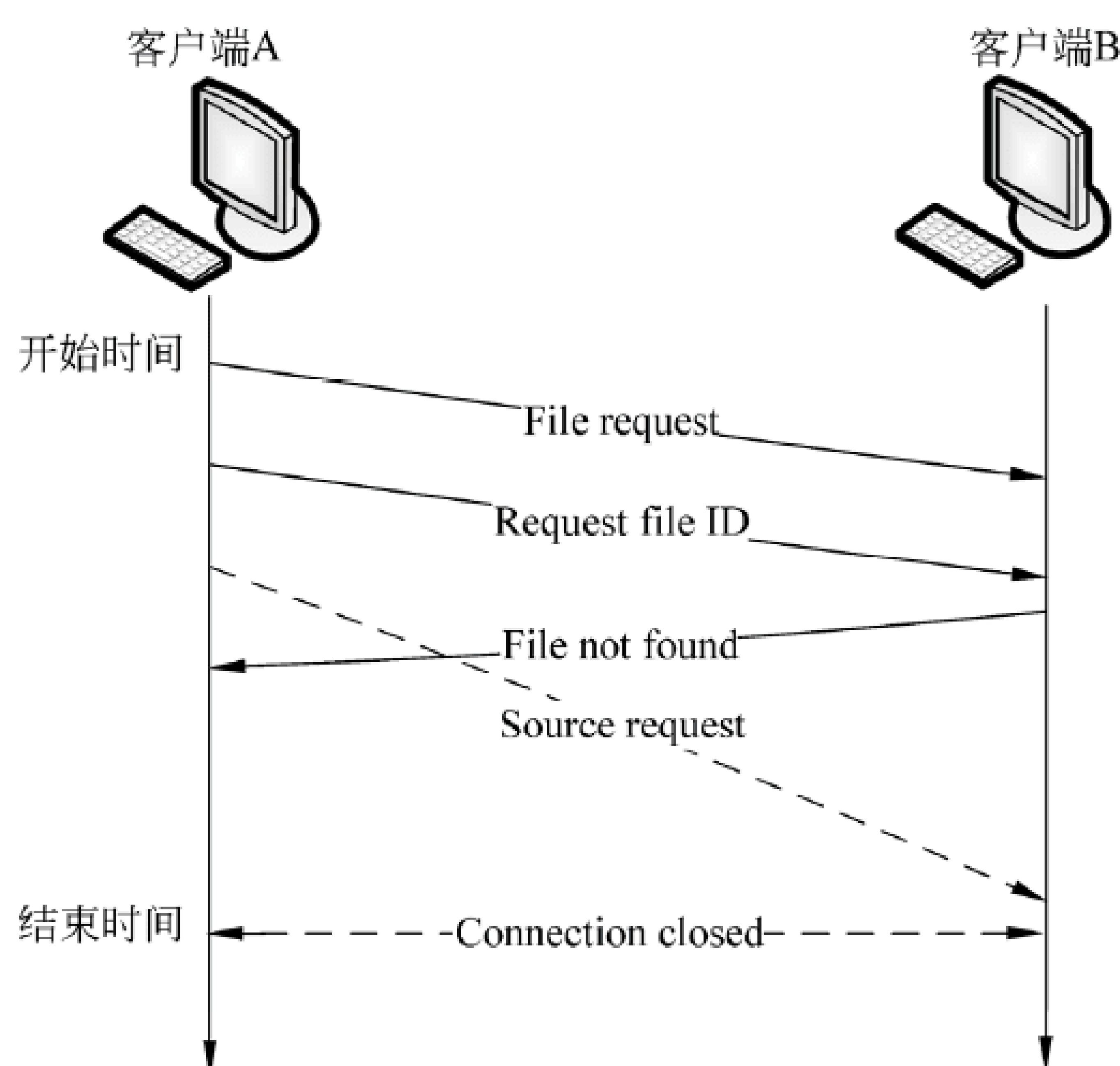


图 7.21　找不到文件时的信息交换过程

（3）加入上传队列

若 B 有请求文件但它的上传列表不为空，意味着有正在下载文件的客户端，也可能在上传队列的客户端 A 和 B 执行完全握手。在这种情况下，当 A 请求 B 开始下载文件时，B 把 A 加到它的上传队列中，回复一个队列级别信息，表示 A 在 B 上传队列中的位置，同时关闭连接。

（4）上传队列管理

对每个需要上传的文件，客户端维护着一个上传优先级队列。队列中每个客户端的优先级是以客户端在队列中的时间和优先级修正为基础计算的，在列头的客户端有最高的积分，积分使用下面的法则计算。

- ❑ 积分=（等级×在队列中等待的秒数）/100 或无穷大。

除了被禁止的用户接受 0 等级外，等级初始值为 100。下载客户端的信用度（范围在 1～10）或下载客户端设置下载文件的优先级（0.2～1.8）都可以改变等级。当客户端的积分比其他开始下载的客户端积分高时，开始下载文件。客户端在以下情况发生时，结束文件下载。

- 上传客户端被用户终止。
- 下载客户端已经得到了它所需文件的所有要下载的部分。
- 下载客户端被其他拥有比它更高优先级的客户端抢占。

注意：为了能让刚刚开始下载的客户端得到少许兆字节数据，以阻止被挤掉，在开始的 15 秒钟内，eMule 会提高客户端初始等级为 200。

（5）到达上传队列列头

当 A 到达了 B 的上传队列的顶部时，B 连接 A，执行初始握手，然后发送同意上传请求消息。那么，A 可以选择或者发送请求块消息来继续下载文件，或者发送取消传输消息来终止下载。

4．数据传输

当客户端之间建立连接，并同意上传和下载文件，客户端之间就进行点对点的数据传输。

（1）数据包

发送和接收文件分块是 eMule 网络活跃的主要部分。发送文件分块要与所有其他 eMule 能承受的数据处理相匹配，其大小在 5000～15000 字节（支持压缩）。为了避免碎片，文件分块信息以块发送，每个块按 TCP 分割成包。在 eMule 0.30e 版中，最大块的大小为 1300 字节（这个数字只与 TCP 的有效负荷有关）。

每个控制信息用单一的 TCP 包发送，有时还包括其他信息，数据信息被分成多个 TCP 包发送。第一个包，包括发送文件分块的头信息，剩下的包仅包括数据。若发送文件部分的大小除以 1300 有余数，那么这个剩余的数和第一个包一起发送（带头信息）。

（2）数据传输顺序

在文件请求回应之后立即开始块传输顺序。下载客户端 A 发送一个开始上传的请求，然后用一个接收上传请求的消息来回应这个请求。A 在这之后立即开始请求文件块，B 通过发送被请求块来回应。

注意：单独的文件块请求可达 3 块之多，所以每个文件块请求可能被可达 3 个发送的块顺序回应。

当两个客户端都支持扩展的客户端协议，文件块可能压缩发送。扩展协议也支持可选的文件信息消息，该消息就在接收上传请求消息之前发送。

（3）选择要下载的块

为了最大化整个网络的吞吐量和共享，eMule 仔细挑选选择块的下载顺序。每个文件被分成 9.28M 的块，每部分分成 180KB 的片。

块下载的顺序是由发送请求文件块消息的下载客户端决定。下载客户端可以在任何给定时刻从各个“源”中下载一个单独的文件块，所有从相同源中请求的片都在同一个块中。

频率规范定义了 3 个区域，即非常稀少、稀少和一般。在每个区域里，标准有特定的权重，用来计算块等级。较低等级的文件分块会被先下载。详细文件的等级范围如下。

- 0～9999：不请求和请求非常稀少的块。
- 10000～19999：不请求稀少和预览块。
- 20000～29999：不请求大部分完成的一般的块。
- 30000～39999：请求的稀少和预览的块。
- 40000～49999：请求的没有完成的一般的块。

这个算法通常选择第一个最稀少的块。然而，部分完成的块、接近完成的也可能被选择，对于通用的块，在不同的“源”之间扩散下载。

5．浏览共享的文件和文件夹

有两个消息流程来处理每对客户端之间的共享文件和文件夹的浏览。第一个是浏览共享文件消息，该消息在初始握手后立即发送。通常由一个浏览共享文件回应消息来回应这个消息。当回应的客户端想隐藏它的共享文件列表时，回应就包含 0 个文件（而不是发送一个指示访问拒绝的消息）。如图 7.22 演示了这个消息顺序。

第二个消息流程随着一个浏览共享的文件夹列表请求开始，通过共享文件夹列表来回应这个消息，然后，对于回应中的每个文件夹，发送浏览共享文件夹内容的消息。当消息到达时，回应的每个消息带有内容列表。如图 7.23 演示了这个消息顺序。

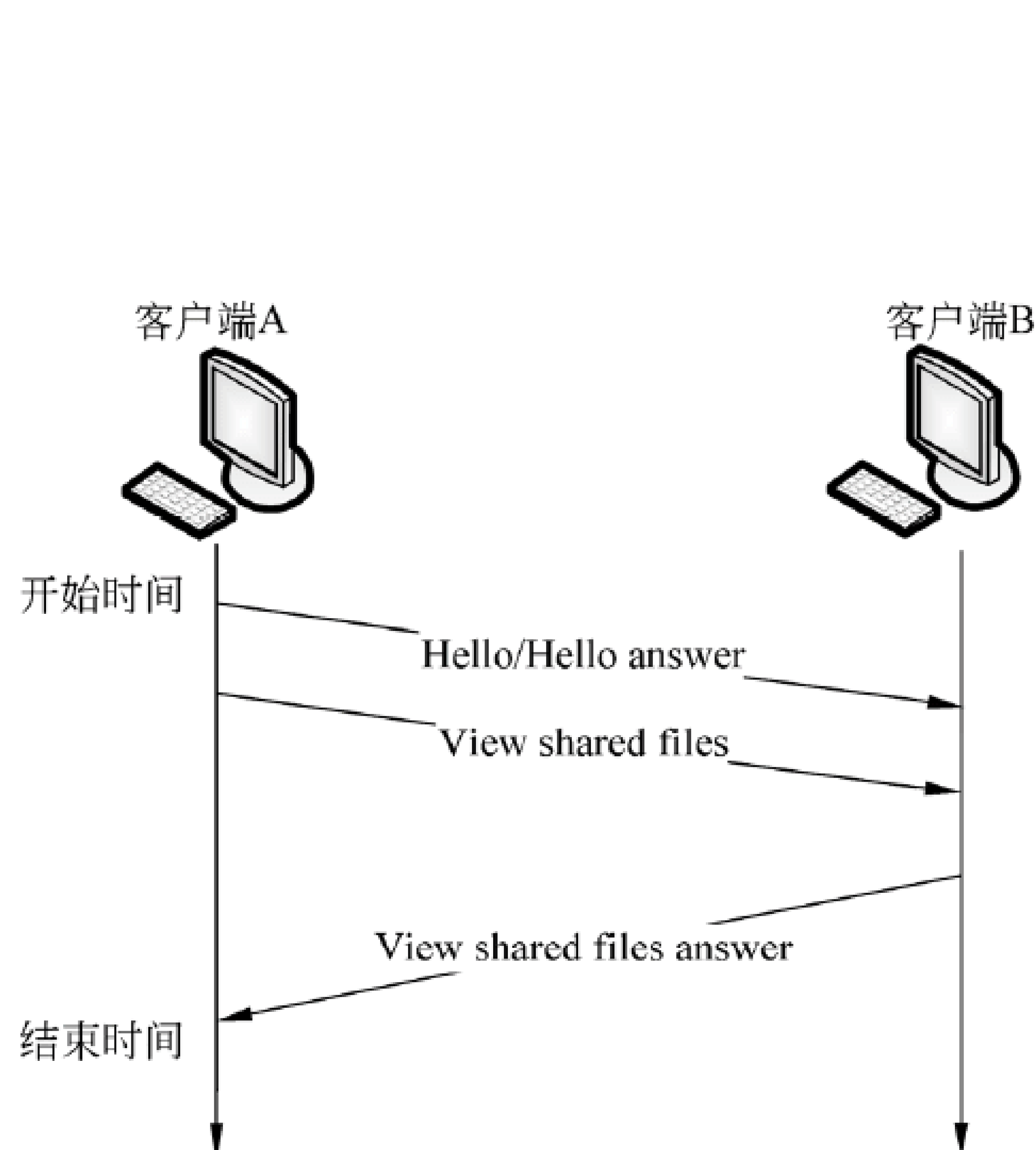


图 7.22　回应无共享文件时消息交换示意图

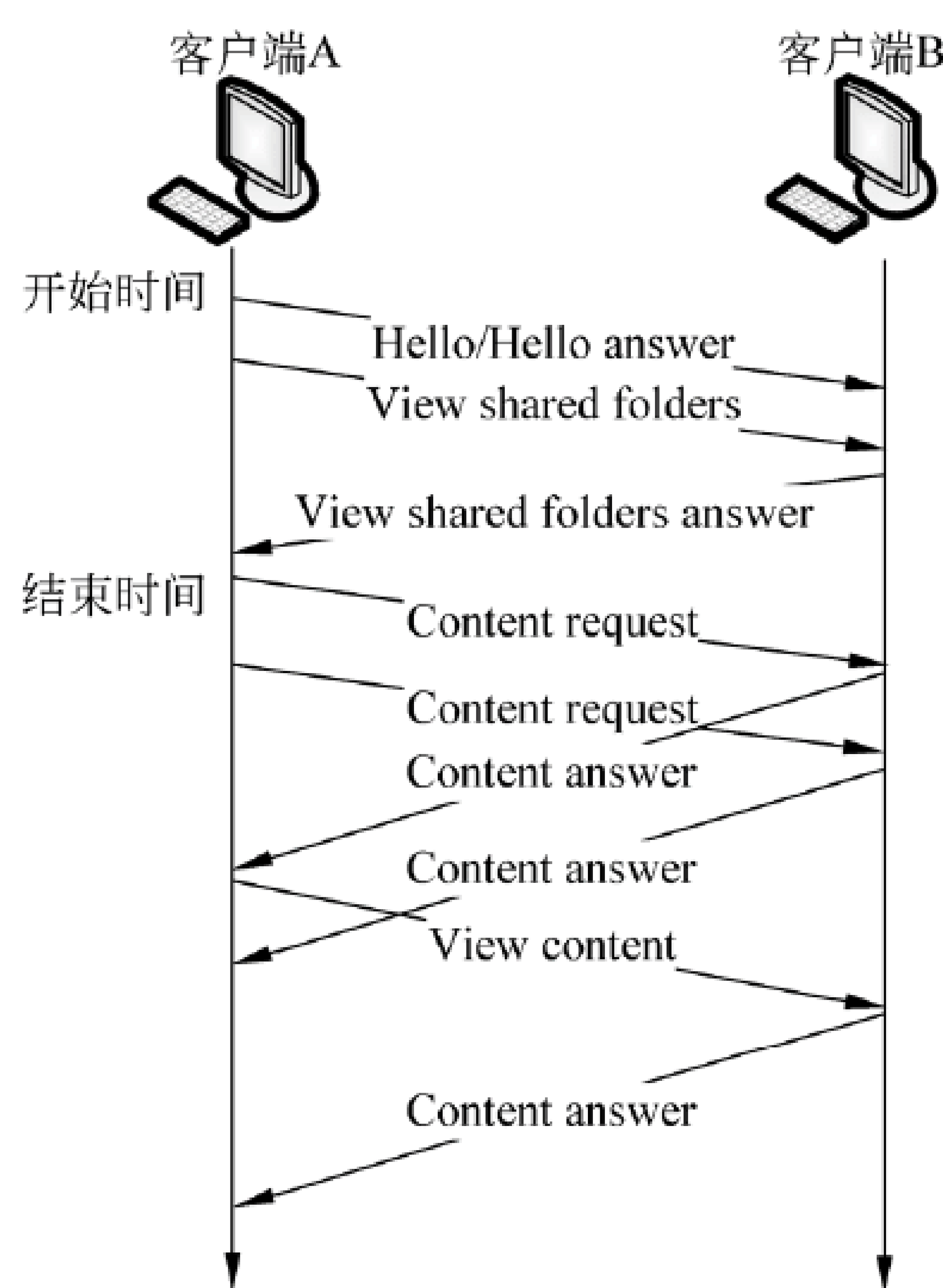


图 7.23　带内容列表的消息交换示意图

6．交换文件分片的哈希

当在搜索列表中选取了想要的文件并开始下载后，eMule 会记录下这个文件的大小、文件名以及一个 MD4 的 Hash 值。这个 Hash 值根据下载文件本身的内容计算得出，表示正在下载的文件是否所指定的，尤其是在文件的其他属性被更改之后（如名称等）这个值

就显得格外重要。eMule 软件得到该信息后，会向所有添加的服务器发出请求，要求得到有相同 hash 值的文件，而服务器则返回持有这个文件的用户信息。eMule 是同时从很多文件提供者那里下载所需的文件，最后再拼成整个文件。

为了取得片的哈希，发送一个哈希集请求，这个请求通过一个包含文件中每块的哈希集回应来回答。如图 7.24 演示了这个信息交换过程。

7．取得文件预览

客户端可以请求对方来获得下载文件的预览。预览是种独立的、随着文件类型不同而不同的应用。eMule 0.30e 只支持图像预览。这个消息交换如图 7.25 描述，只包含两种消息，预览请求和预览回应。

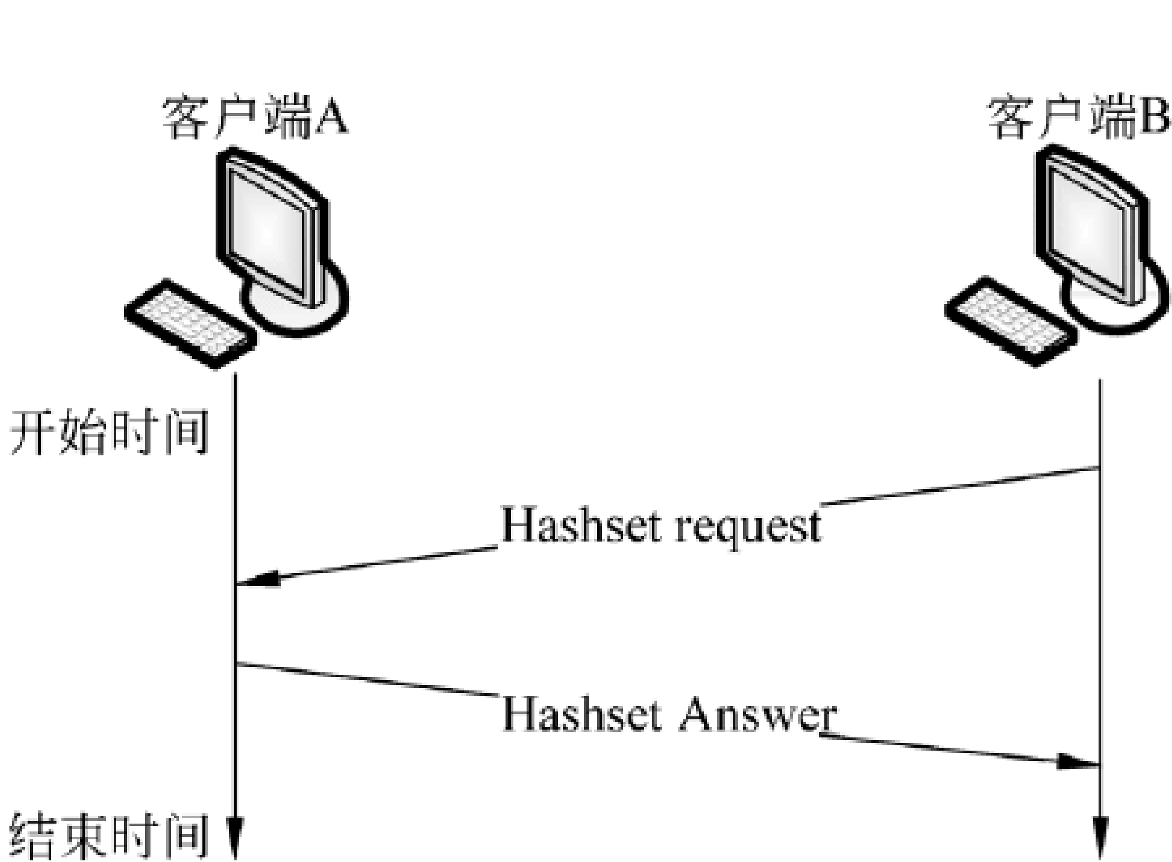


图 7.24　取得分片哈希的消息交换过程示意图

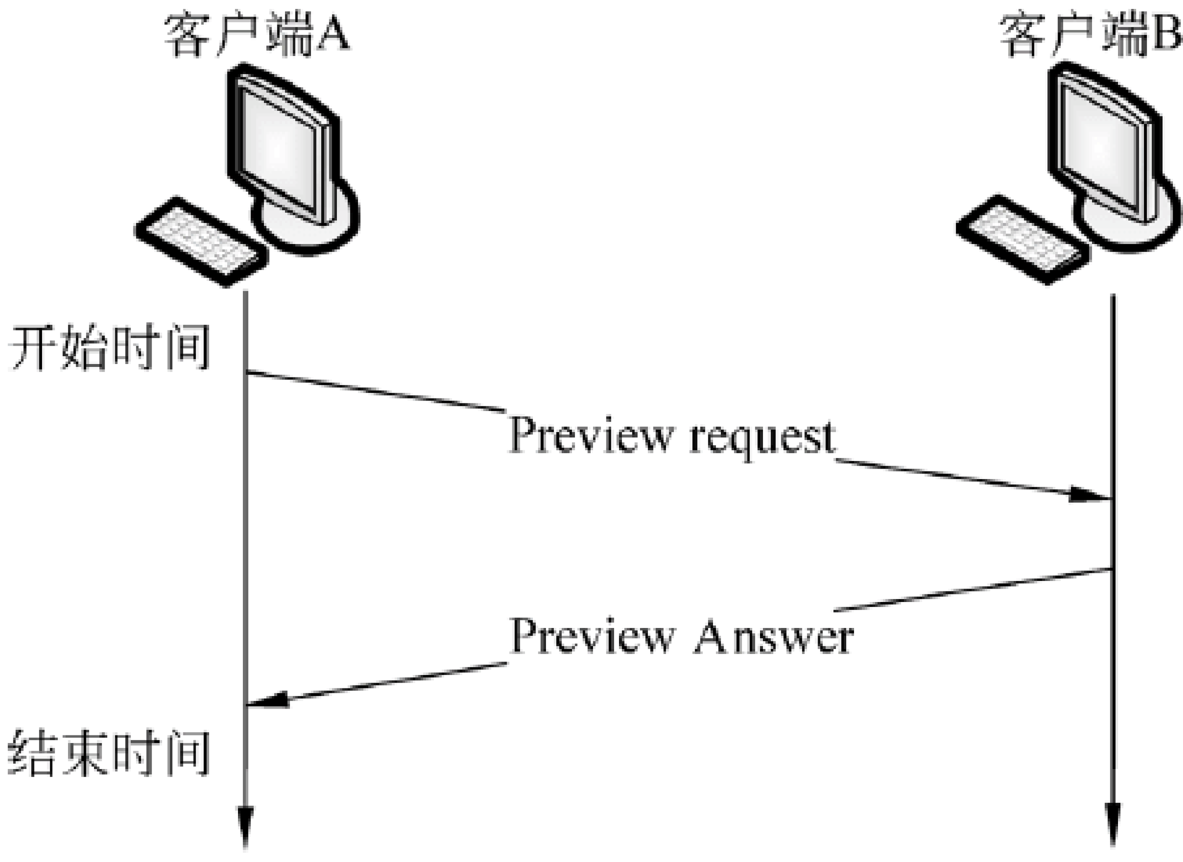


图 7.25　取得文件预览的消息交换过程示意图

7.3.7　eMule 协议分析之——客户端到客户端的 UDP 连接

eMule 客户端周期性地用 UDP 协议发送消息。在 eMule 0.30e 中，使用的 UDP 消息只是询问客户端在对方下载队列中的位置。这个简单的请求-应答方案是随着重复要求文件消息而开始的。对于这个请求，有 3 种可能的回应。

- 队列等级：客户端在发送者队列中的等级。消息交换过程如图 7.26 所示。

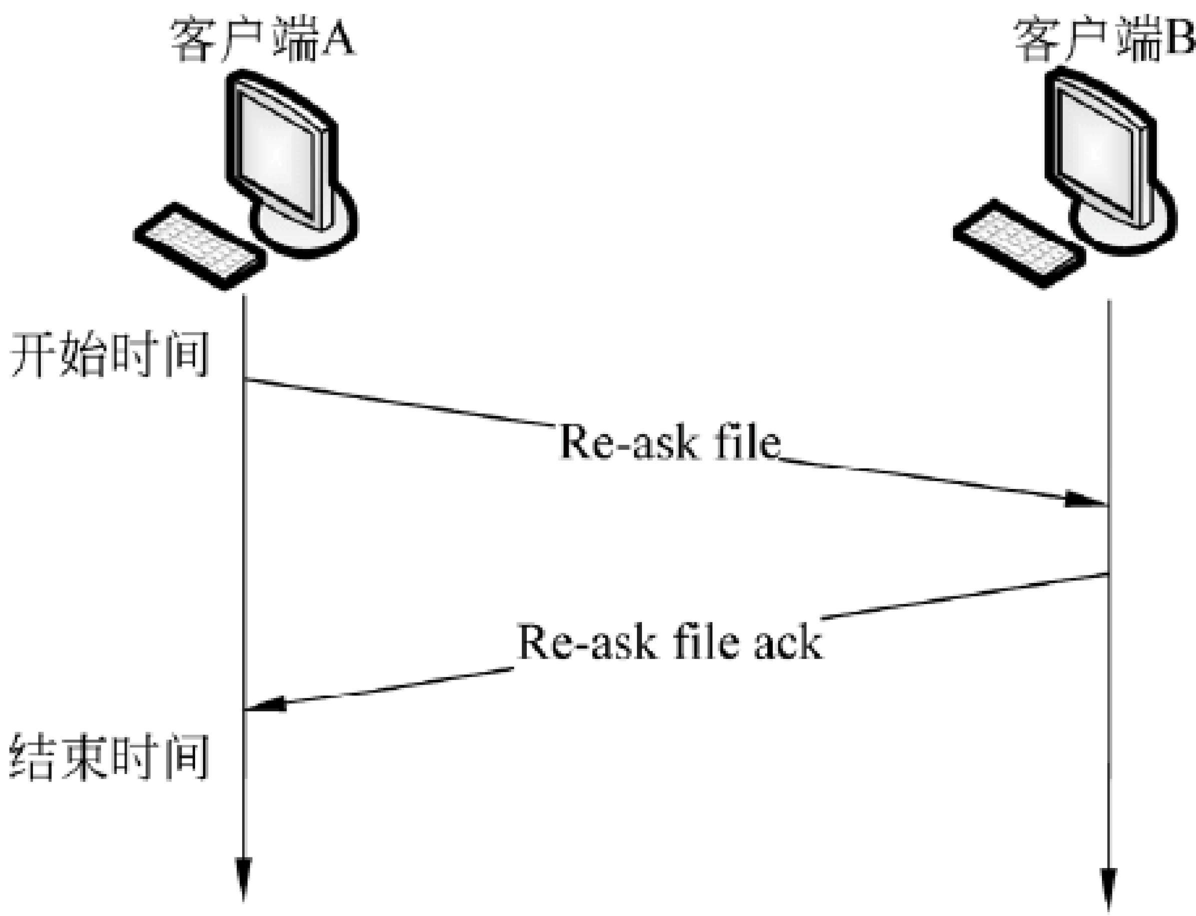


图 7.26　回应队列等级的消息交换过程

- 队列满：发送者队列已满。消息交换过程如图 7.27 所示。
- 找不到文件：发送者在它的列表中没有被请求的文件。消息交换过程如图 7.28 所示。

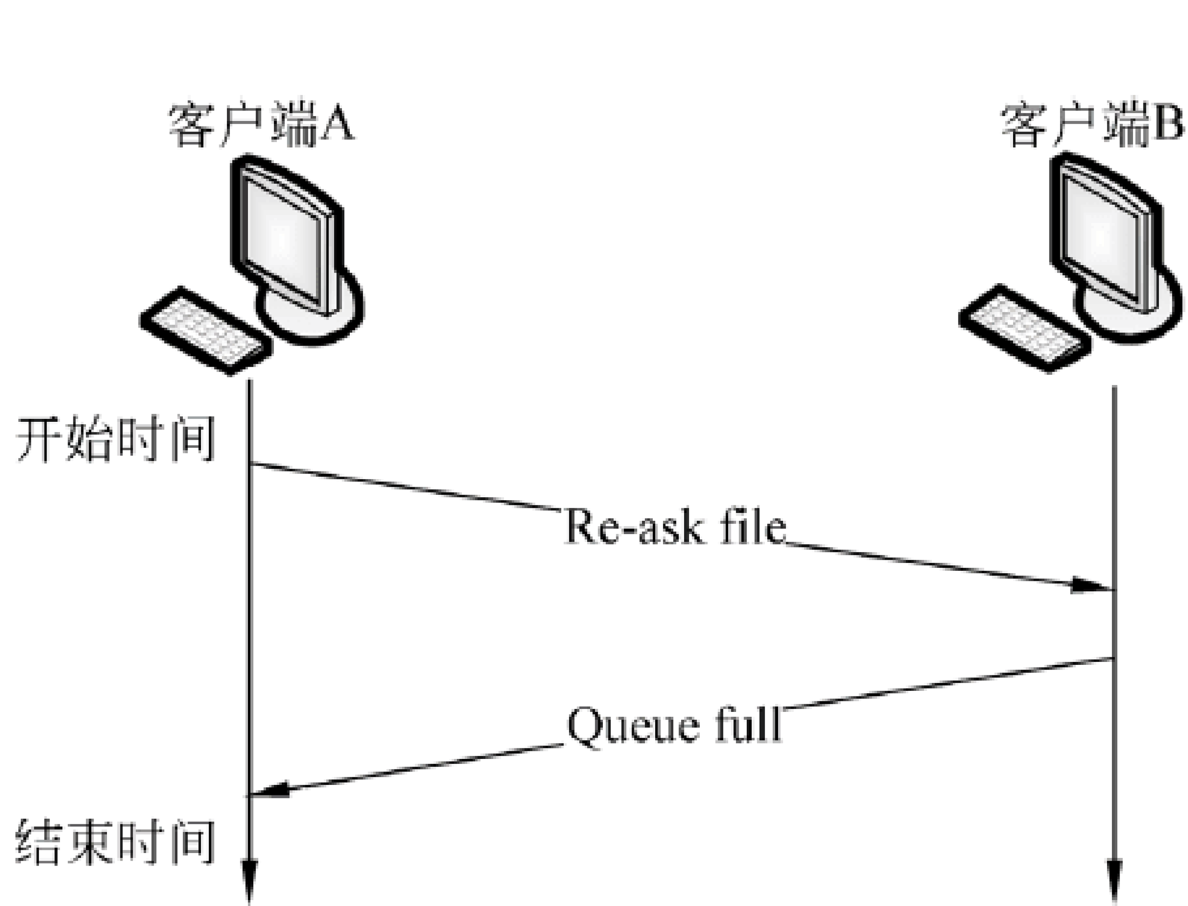


图 7.27　队列已满时消息交换的过程

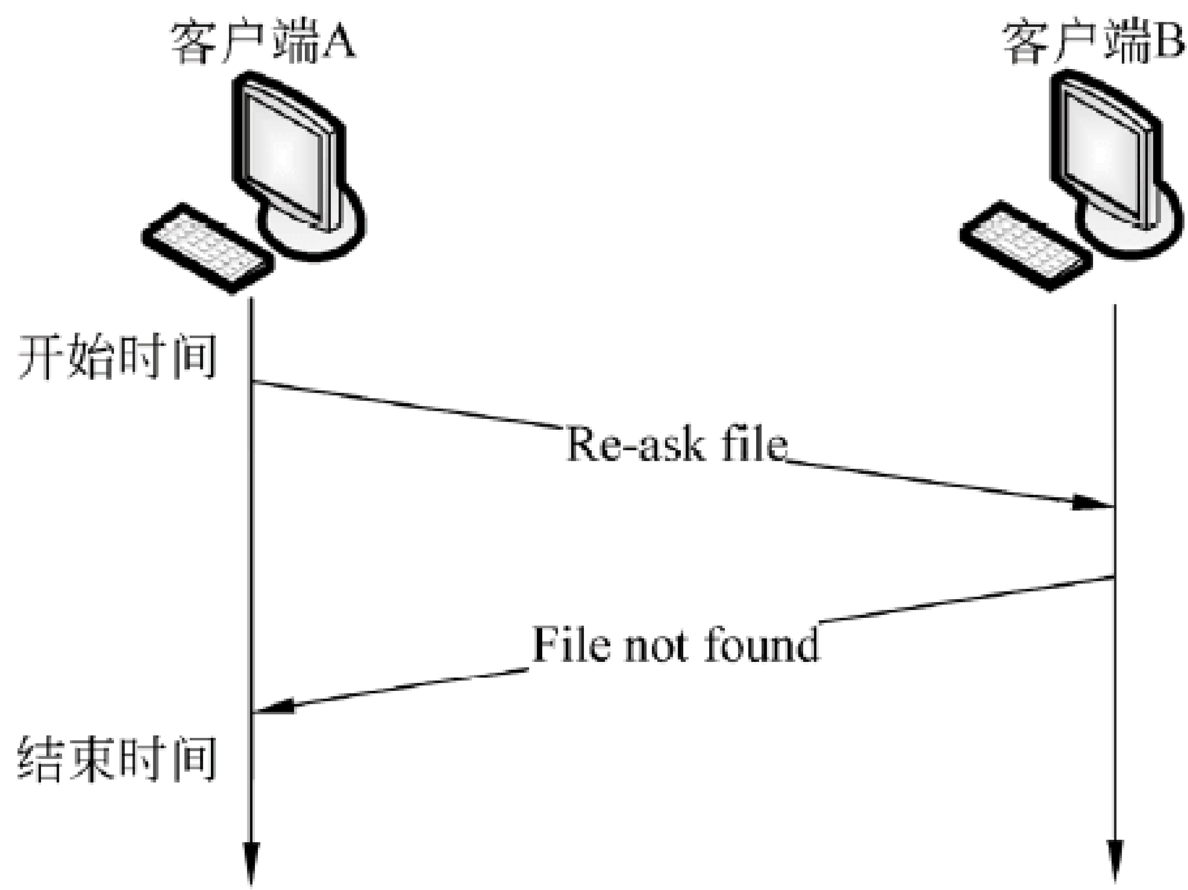


图 7.28　文件没找到时回应消息的过程

重复要求文件消息大约每 20 分钟的间歇发送到每个客户端，这些客户端把发送者加入到它的下载队列中。

7.4　eMule 知识进阶

上文对 eMule 的基本知识和 eMule 协议的重点内容进行的介绍，本节将带领读者进一步学习 eMule 的一些重要概念和特性。这些概念包括 eMule 的 eD2k 的链接、eMule 的高 ID 和低 ID 的由来，以及 eMule 里的信用积分系统等。理解这些特性和概念对深入学习和掌握 eMule 有重要的作用。

7.4.1　eD2k 链接的含义

eD2k 链接是一个特别的链接格式，允许直接加入一个下载到 eMule。这些链接允许 eMule 直接从 Web 网页下载，并且使它可以容易地在互联网络中交换下载。eMule 中的 eD2k 链接有很多种形式，不同的链接形式其格式也不尽相同。

1．eD2k链接的格式及说明

- 基本的 eD2k 链接：链接格式 eD2k://|file|<文件名称>|<文件大小>|<文件哈希值>|/。一个 eD2k 链接包含必要的文件描述项有名称、大小及哈希值形成基本的格式。
- eD2k 片段哈希值链接：链接格式 eD2k://|file|<文件名称>|<文件大小>|<文件哈希值>|p=<片段哈希值>|/。文件的完整片段哈希值确保文件总是正确地并且帮助新的罕见的文件散布。
- eD2k 来源链接：链接格式 eD2k://|file|<文件名称>|<文件大小>|<文件哈希值>|/|sources，<IP:端口>|/。加入一个或多个已知的 eMule 来源再格式到这个链接，

提供立即来源来下载。

- eD2k 主机链接：链接格式 eD2k://|file|<文件名称>|<文件大小>|<文件哈希值>|/|sources，<主机名称:端口>|/。相同于来源链接但使用主机名称来替代 IP。特别是在变动 IP 提供更灵活的。一个主机名称必须设定在选项->扩展->自己的 eD2k 链接主机名称。
- eD2k HTML 链接：链接格式<a href="eD2k://|file|<文件名称>|<文件大小>|<文件哈希值>|/">显示在 Web 网页名称，这种 HTML 的链接格式，很容易建立一个链接来显示在一个 Web 网页上。
- eD2k HTTP 来源链接：链接格式 eD2k://|file|<文件名称>|<文件大小>|<文件哈希值>|s=http://anyweb。net/文件名称|/。eMule 也能够直接从 Web 来源下载。一个对于 Web 管理员非常有用且方便的格式，见下文的 Link Creator 以了解更多信息。
- eD2k 根哈希值链接：链接格式 eD2k://|file|<文件名称>|<文件大小>|<文件哈希值>|h=<根哈希值>|/。根哈希值链接允许由 AICH 提供一个可靠的值来做进阶错误修正及检查的方式。

2. 在eMule建立链接

在 eMule 中，非常容易产生每个文件的 eD2k 链接，除了 HTTP 链接之外。在 eMule 系统的文件列表中选择任一个下载右击，如图 7.29 所示，就能显示出相应的 eD2k 链接。

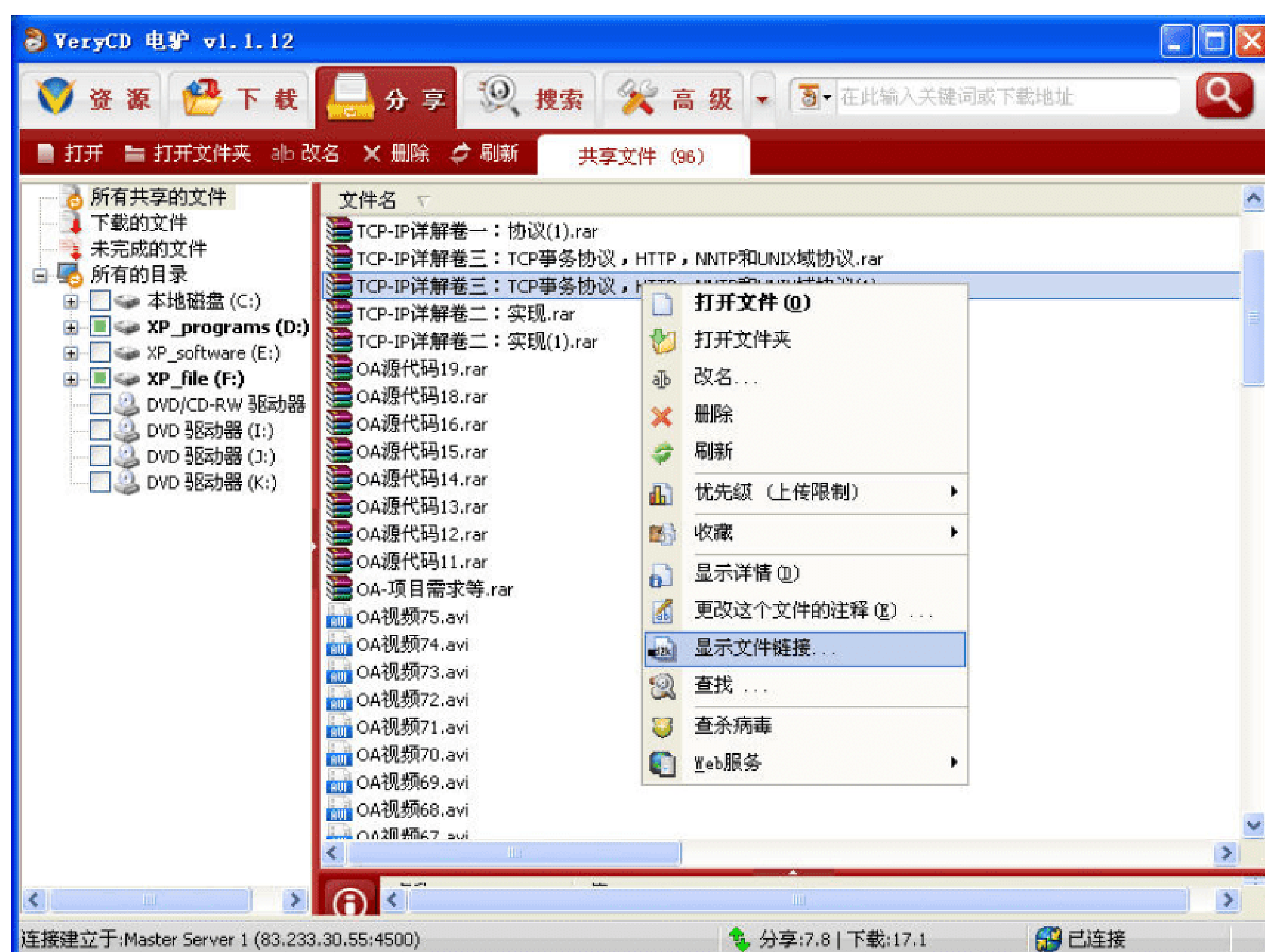


图 7.29　显示 eD2k 链接的操作方法示意图

对于已共享文件，经常需要将共享文件的 eD2k 链接发布出来，抽取已共享文件的 eD2k 链接的方法如图 7.30 所示。

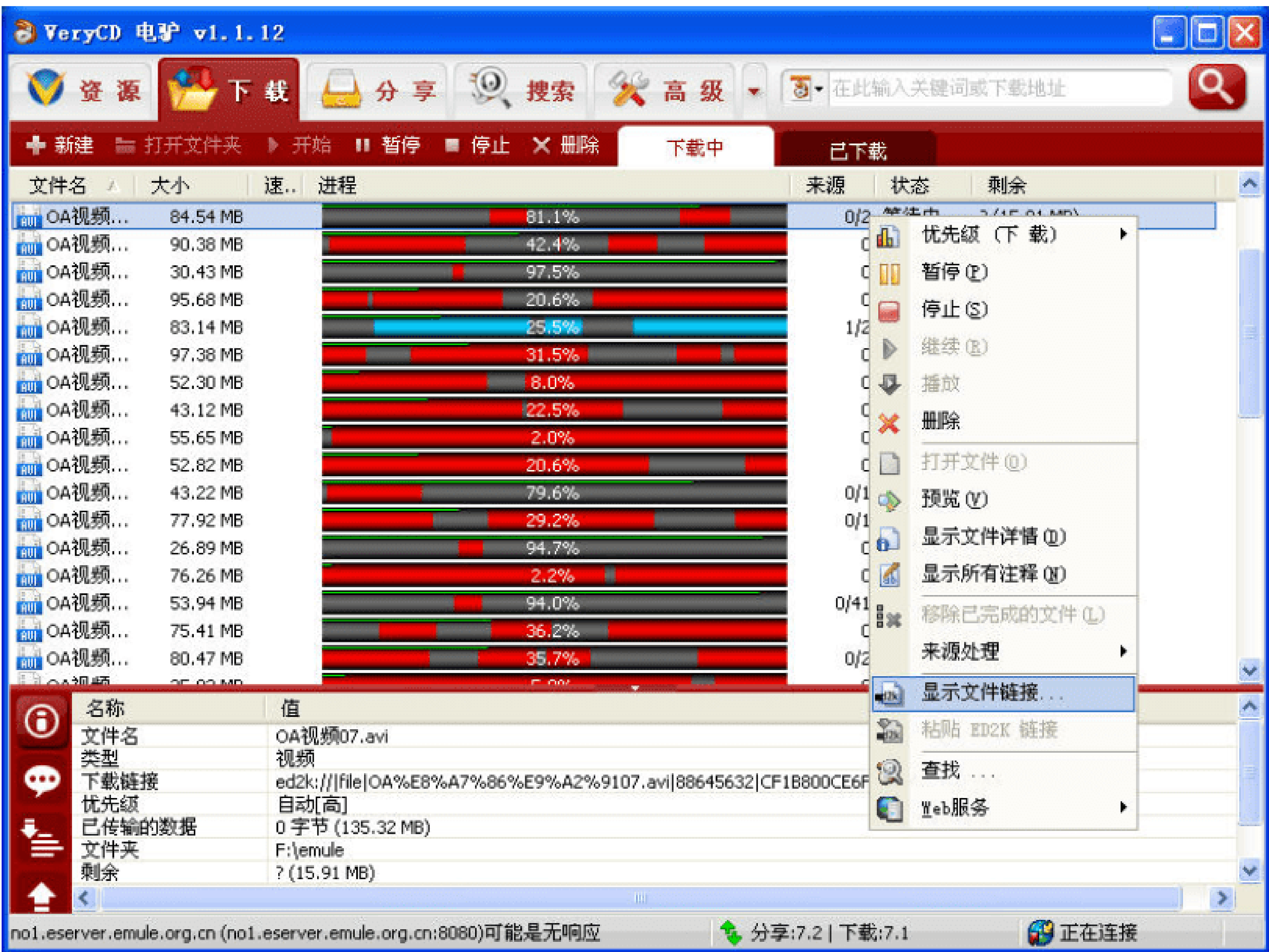


图 7.30　已共享文件的 eD2k 链接的显示方法

3．用Link Creator制作eD2k链接

Link Creator 是 eMule 的一个额外附加工具，可以方便地产生各种格式的 eD2k 链接，特别是还可以方便地建立 HTTP 来源的链接。

Link Creator 的使用界面如图 7.31 所示，通过 Link Creator 工具，可以是非常容易地、快速地产生这样的链接。

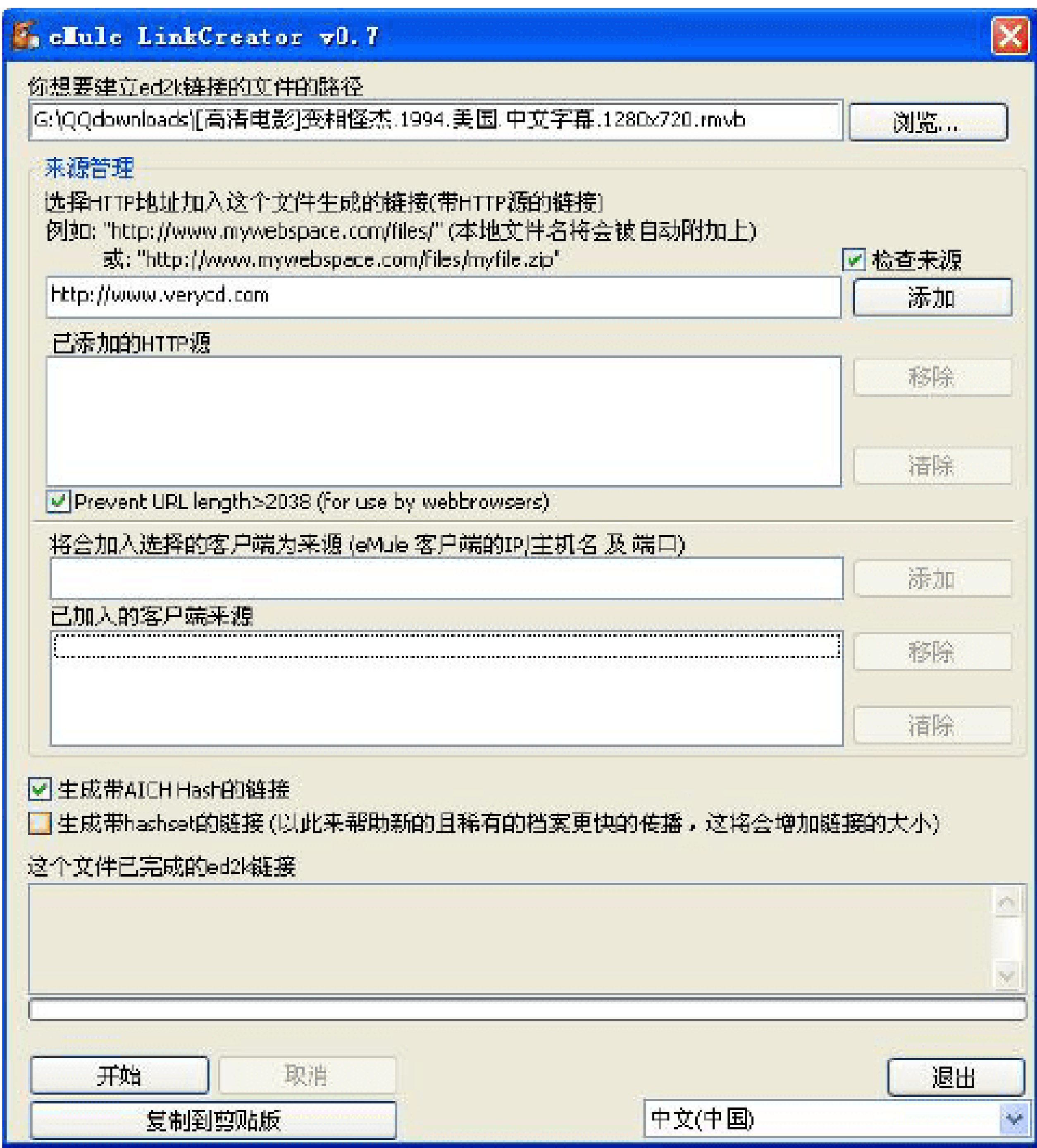


图 7.31　Link Creator 制作 eD2k 链接的界面截图

在图 7.31 所示的操作界面中，只需输入文件来计算哈希值并且产生链接。选择性的 HTTP Web 来源也许也能被加入。使用“移除”按钮可移除一个已选择的 HTTP 来源，或使用“清除”按钮移除全部。

建议加入片段哈希值，这样，它可以改善下载的可靠性。单击“开始”按钮就可以产生相应的链接了。

7.4.2　eMule 系统中的 LowID 与 HighID

在本书的第 5 章中提到过 eMule 的 High ID 与 LowID 之分，在系统地学习过 eMule 相关知识后，再介绍一下这个知识点，以加深读者对这一概念的理解。

1．为何eMule会有HighID、LowID的之分

这就要从互联网结构说起了，不过长话短说——HighID/LowID 不是 eMule 这个软件的缺陷。

High ID 就是拥有独立公网 IP 并且能提供端口（默认 4662）给 eMule 工作的用户，此类用户可以和任何 eMule 兼容客户端连接与下载。

Low ID 一般都是没有公网 IP 的内网用户，两个 Low ID 用户之间是无法直接连接的，所以 Low ID 的用户下载源会相对少一点（无法从其他 Low ID 用户那里下载）。

无论什么类型的 P2P 软件，MSN、QQ、BT 等，都无法实现两个内网用户——LowID 用户之间的直接通信。MSN、QQ 等即时聊天工具都是通过服务器转发数据实现的，像 MSN，所有的数据都是通过服务器转发，因此 MSN 上是看不到好友的 IP 地址的。而如果你用过早期 QQ 版本的话，应该知道有一种情况经常会发生，就是即使聊天的 2 个人都在线，所有的聊天内容后面都会跟上“通过服务器转发”，现在的版本只是已经去掉了这句话。

然而，为传输大量数据而设计的 eMule、BT 等 P2P 软件，显然无法使用“服务器转发”来实现 LowID 用户之间的通信——因为大量的数据转发会无端浪费极大的网络带宽，并且 eMule、BT 之类免费软件也无法提供要求如此之高的服务器来完成这项任务。BT 之中没有 LowID 的说法只是因为它不提，而不是它能够解决这个问题。所以不要说因为 LowID 的关系而放弃 eMule 转用 BT，那是荒诞无稽的笑话:-）。

2．为什么要向LowID说不？

从目前的情况来看，国内是 LowID 的骡子占大多数，也就是这大多数用户之间无法通信，所以成为一个 HighID 就显得非常的必要。不但可以让自己获得更多的下载机会，还可以在众多 LowID 之间起到中转的作用，帮助他们下载。下面将会提供足够多的信息，帮助获取 HighID。

两个内网用户无法直接连接是现有的 Internet 协议造成的，不要把它理解为 eMule 的缺点，任何软件都无法帮助两台内网的计算机直接连接。

注意：一般的即时通信软件都通过服务器中转来解决这个问题，而 eMule 中的 HighID，其实也起到了为 LowID 们中转的作用。

如果连接服务器得到的是 LowID，一般有两种可能：一种是确实是内网用户，正常途径无法得到 HighID；还有一种就是其实是公网用户，但是由于防火墙的限制，使你拿到了个 LowID。

3．eMule中关于公网、内网、NAT的几个知识点

公网和内网是两种 Internet 的接入方式。

（1）内网接入方式：上网的计算机得到的 IP 地址是因特网上的保留地址，保留地址有如下 3 种形式。

- 10.x.x.x；
- 172.16.x.x 至 172.31.x.x；
- 192.168.x.x。

内网的计算机以 NAT（网络地址转换）协议，通过一个公共的网关访问 Internet。内网的计算机可向 Internet 上的其他计算机发送连接请求，但 Internet 上其他的计算机无法向内网的计算机发送连接请求。

（2）公网接入方式：上网的计算机得到的 IP 地址是因特网上的非保留地址。公网的计算机和 Internet 上的其他计算机可随意互相访问。

NAT（Network Address Translator）是网络地址转换，它实现内网的 IP 地址与公网的地址之间的相互转换，将大量的内网 IP 地址转换为一个或少量的公网 IP 地址，减少对公网 IP 地址的占用。NAT 的最典型应用是：在一个局域网内，只需要一台计算机连接上 Internet，就可以利用 NAT 共享 Internet 连接，使局域网内其他计算机也可以上网。使用 NAT 协议，局域网内的计算机可以访问 Internet 上的计算机，但 Internet 上的计算机无法访问局域网内的计算机。

Windows 操作系统的 Internet 连接共享、sygate、winroute、unix/linux 的 natd 等软件，都是使用 NAT 协议来共享 Internet 连接。

所有 ISP（Internet 服务提供商）提供的内网 Internet 接入方式，几乎都是基于 NAT 协议的。

7.4.3　eMule 的积分系统

eMule 的积分系统就是为了鼓励用户尽可能多的上传文件，真正履行 eMule“人人为我，我为人人”的理念。

使用 eMule 下载的时候，获取的所有资源都源于其他网友，如果每个人都只求索取而不谈奉献，那么 P2P 下载只能走向死亡，同样 eMule 也就没有用武之地了，幸运的是，eMule 的开发者意识到了这一点，为了鼓励那些上传者，eMule 在目前的版本中都包含了一个信用系统，上传量大者可以得到较高的信用积分，从而得到更多的下载机会、更快的下载速度。

1．eMule的积分系统

在使用 eMule 下载的时候经常看到，明明有很多的源，却没有下载，这是由一个积分机制在控制着。eMule 系统会根据用户的积分情况形成队列，队列表里是正在等待上传的

网友们。不同的用户有着不同的得分，也决定了你在队列里的优秀次序。这样，如果你的积分不够，你就会被排在底端，尽管有很多“源”但也无法得到更快的下载速度。

2．eMule的信用系统（Credit System）

信用系统是用来回报那些为 eMule 网络做贡献的用户，如那些经常上传文件的用户、经常向其他的 eMule 客户端提供下载源的用户。

eMule 中严谨的队列系统构建于用户在队列中的等待时间。信用系统为这个等待时间提供一个比例，这个比例将两个用户间的上传、下载大小考虑在内。一个用户给另一个用户上传的越多，他在这个用户队列排名上升就越快。比例由两个用户间传输的数据大小计算得来。使用的数值可以在用户的详细信息对话框看到。要查看信用信息，右击用户，在弹出的快捷菜单中可选择查看详细情况。

3．信用系统设置

在 eMule 客户端软件中，依次打开“选项”|“扩展设置”选项，在这个对话框里可以设置启用信用系统——Credit System（受益上传者），如图 7.32 所示。

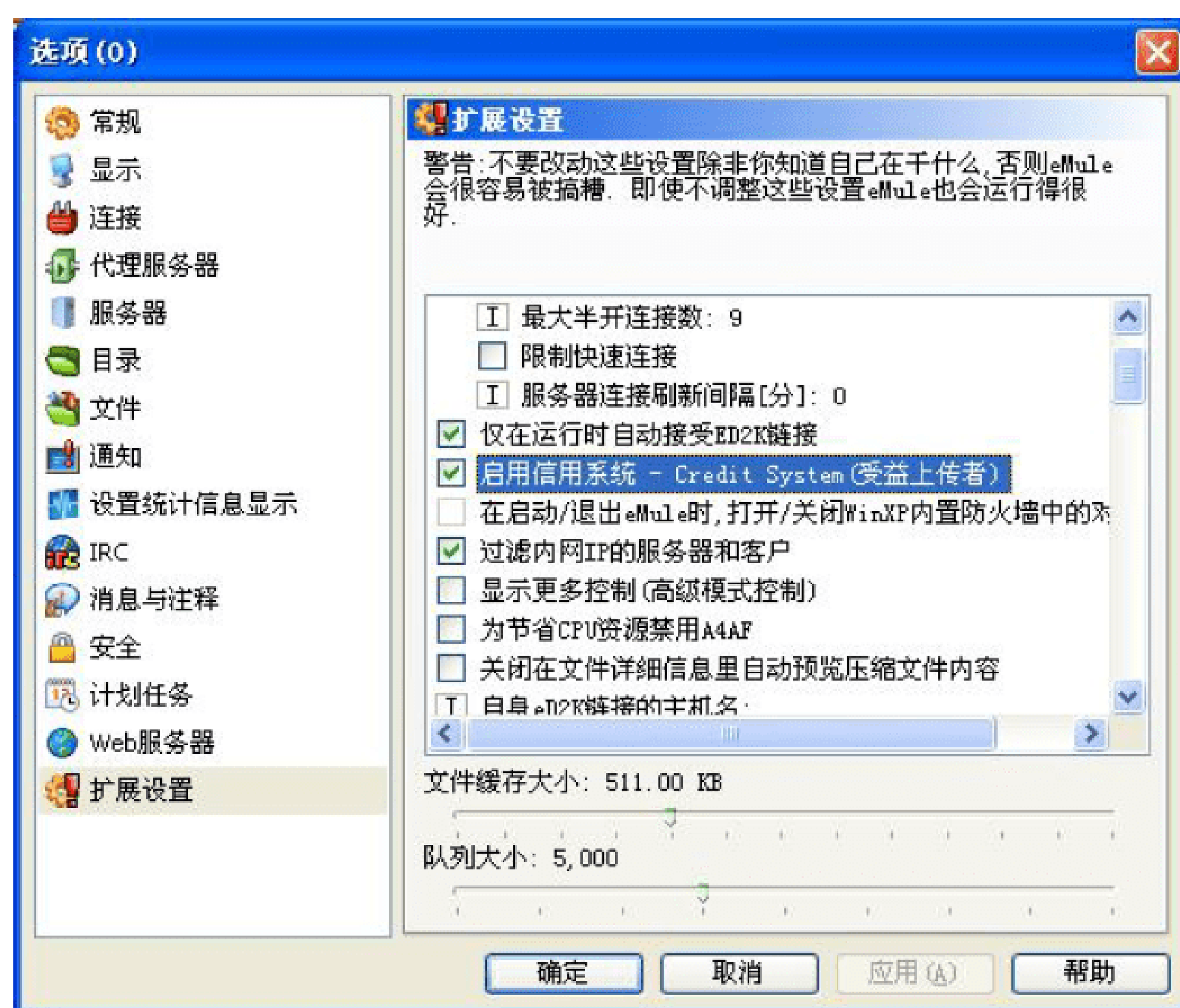


图 7.32　eMule 中信用系统的设置界面

注意：*以上是 eMule 0.49c 版的设置情况。*

如果你启用了这个 Credit System，那你的 eMule 在 5 个月后就会出现客户被删除的信息。那这个信用系统（Credit System）是用来让上传者受益的。举个简单的例子，比如 A 给 B 上传了，那么 B 就会记下 A 的 User Hash。如果下次 A 要下载 B 的东西时，B 就会给 A 的评分比普通的用户高。这样 A 在 B 这儿就可以少排队或者不排队进入下载。

当然，A 和 B 建立的这种信用的评分关系只能持续 5 个月，如果 5 个月内两个用户都

没有再建立过连接，就会出现上面被删除的情况。这也就是为何在使用 eMule 时要加大上传原因。因为上传多了后，带来的好处就是以后下载东西时可以少排队或者不排队。

所以，eMule 用户应该对自己的 UserHash 要加倍的珍惜，特别是上传量大的朋友，经常备份 config 下的文件。一旦 UserHash 变了，你和其他的 eMule 客户端建立的这种信用关系也就没有了。

7.4.4　如何架设一个 eMule 服务器

eMule 服务器（eD2k 服务器）在 eMule 的整个运行过程中有着非常重要的作用，尤其是在找“源”与文件搜索方面，eD2k 服务器暂时无法被 KAD 完全取代。eD2k 服务器可以鉴别虚假与恶意文件，维系 eMuel 与其他不支持 KAD 协议的 eD2k 兼容客户端之间的联系。有了 eD2k 服务器，eMule 才能更好地运行。

注意：eMule 网络是由 Kad 网络和 eD2k 网络构成的，而 Kad 网络拓扑完全不需要服务器，所以所谓的 eMule 服务器，其实就是 eD2k 服务器。

最早编写 eD2k 服务器软件的是创立 eD2k 协议的 MetaMachine，称为 DSERVER，版本一直发展到 16，然后转交给法国的 Lugdunum 小组维护，现在发展到最新的 17.15。Lugdunum 也称 eserver。目前已知的所有知名 eD2k 服务器，包括 eDONKEY Server、TVU、UseNext 在内全部基于 Lugdunum 运作。

注意：Lugdunum 一词的原意是法国城市里昂的旧称，在这里指的是目前全球最流行的架设 eD2k 服务器的应用软件，免费但不开源。可通过搜索引擎或是 Lugdunum 的相关信息并下载最新版本的 Lugdunum 软件进行试用。

Lugdunum 主要设计用于 Linux 平台运作，支持多核心与 64 位运行，但也有面向 WIN32、solaris 与 FREEBSD 平台的版本。

注意：Lugdunum 17.15 的 Win 32 版已经被证实无法正常运作，不论客户端的情况如何，与服务器连接时，所有连接上的客户端均会被分配为 LowID，所以在 Windows 下，基本无法使用 Lugdunum。

下面就讲一下，如何在 Linux 环境下配置一个 eD2k 服务器。

1．准备工作

- 有一个 Linux 的系统环境，笔者试验的 Linux 环境是 Ubuntu9.04。

注意：Linux 的安装、使用及软件安装请读者参考其他相关资料。

- 下载与你的系统和 CPU 对应的 Lugdunum 软件，然后解压到任意文件夹中。
- 下载 ip-to-country.csv 与 ipfilter.dat，把它们放在 Lugdunum 存放的位置。
- 编写 Lugdunum 的配置文件——donkey.ini。

注意：Lugdunum 的配置文件是整个服务器架设的核心部分，eD2k 服务器的每一个变量都被写在这个文件里。eserver 开始运行时会读这个文件。Lugdunum 不能接受语法错误，否则它就不会工作，或者它会忽略某些写得不对的命令（以及后面的任何东西）。大小写也是非常重要的。所有单独条目均不能有注释。

2. 关于donkey.ini文件的配置

一个完整的 donkey.ini 配置文件的格式大致如下，类似于一个<key，value>值对的形式存在，“＝”的左边是 Key 值，表示要设置的命令，“＝”的右边是 Value 的值，表示“Key”要取得值。本例中都用？号代替，表明是将要设置的值。

```
[server]
name=?
desc=?
thisIP=?
port=?
seedIP=?
seedPort=?
verbose=?
public=?
threads=?
type=?
tableSize=?
maxClients=?
minVersion=?
maxVersion=?
logFile=?
welcome[0]=?
welcome[1]=?
welcome[2]=?
```

以上是一个空的 donkey.ini 文件的格式，现在对这个文件中的命令逐一讲解并对其赋值的情况进行说明。

注意：以下的所有命令、符号都是在英文状态下输入的，尤其是“=”号两边没有任何空格。

[server]

这个命令启动控制进程。它必须被写在方括号里。在最后的那个方括号后面没有空格，这一点非常重要。否则将使服务器无法理解。

name=eMuleServer_Test

服务器的名字，就像在 eMule 客户端看见的那个服务器的名字一样。可以用任何想用的文字各数字的组合来描述这个名字，这里就简单的取名为 eMuleServer_Test。

desc=http://www.p2p.book

服务器描述信息，简单地说就是用来描述这个服务器的特征、用来干什么的、有什么特色等。可以用任何文字和数字的组合来描述。

🔔注意：这里虽然说可以用任何文字或数字组合，但尽量不要使用一些特殊字符，否则容易引发一些不可知的错误。

thisIP=123.115.215.7

thisIP 这个词要注意大小写，这里给定的 IP 地址只是一个例子，这个配置的意思是，这将告诉服务器你自己的、公开的公网 IP。程序开始时，服务器会把它工作的 IP 发送到其他的服务器。而 eMule 用户也会通过你提供的这个 IP 连接到你的服务器。

port=4668

在这里要指定可以跟服务器建立联系的端口。这个端口的值可以有两种设定：

- ❑ 此条目空着，不设置任何值。
- ❑ 自定义设置一个端，如本例中设置端口值为 4668。

首先，这个端口指的是 TCP 端口，在 Lugdunum 中，一般情况下这个端口是 TCP：4661，如果想用这个 4661 的端口，那么 port 命令的设置就可以空着，系统会认为此端口就是 4661；如果由于某些技术原因该 4661 端口不可用，则可以用其他的端口。此时就必须设定 port 的值了。

🔔注意：如果可能的话，尽量不使用 4661，因为某些 ISP 为了阻止 eMule 占用流量，封锁了 4661 与 4662 端口，所以尽量不使用默认的端口。

seedIP=62.241.53.16

注意 seedIP 的大小写！这里的 seed 可理解为网络上的其他正在运行着的 eD2k 服务器，也叫种子服务器。而 seedIP 就是种子服务器的 IP 地址，设定此值就是告诉将你的这个服务器，可以从哪里连接到网络上，设定了种子服务器的 IP 地址，你的服务器在启动运行后才能找到它。

这个种子服务器，准确地说应该是你的服务器首先与之建立连接的那个服务器，你的服务器在启动以后会发送自身的 IP 地址和端口号。建立连接后，就会获得种子服务器所知道的所有 IP 地址。

seedPort=4668

注意大小写！这是设置种子服务器的端口号，同上面设置的 port 有类似的意思，种子服务器的默认端口也是 TCP：4661，此条目设置为空的时候就取默认值；如果“种子服务器（Seed-Server）”的端口号不同于默认的 TCP：4661，就必须设定一个条目的值。这里设置的值是 seedPort=4668。

logFile=true/false

注意大小写！关于服务器日志文件的相关配置，就是是否把日志信息显示出来，Lugdunum 的日志信息有以下 3 种显示方式：

（1）可以直接把日志信息写到硬盘里，而无须将这些信息直观地输出到服务器的显示界面上，在此情况下，就要把显示日志输出信息的功能关闭。如果不想显示输出而想用日志文件的话，设置 logFile=true。

（2）Lugdunum 日志信息的另一种显示方式就是，把所有的日志信息直接显示输出到 Lugdunum 服务器的屏幕上，此时设置 logFile=false。

（3）还有一种方式，就是兼顾（1）和（2）既把日志信息从显示的屏幕进行输出，同

时也把日志文件写到本地硬盘中，但强烈建议不要使用这种设置，否则会给主机带来很大的负担。

logFile 的默认设置是 false——让日志信息在屏幕显示输出即可。

verbose=true/false

verbose 本身是“啰嗦的、冗长”的意思，主要用来控制系统注意信息的显示。

在系统运行过程中，如果想看所有的系统显示的注意信息，应该把 verbose 激活，即将它设成 verbose=true。如果设置了 verbose=false，一些系统的注意信息将不会被显示，但某些错误信息仍旧会显示出来。例如，服务器仍然会打出 ERROR: unknown type MetaTag::MakeTag() 72 或者类似的错误信息。

出现这些错误信息的时候，一般都是一些警告性的，不会影响服务器的工作。如果服务器出现问题的时候，这些信息将是很重要的参考。Verbose 默认值是设成 false，也就是关闭注意信息的显示。

注意：Lugdunum 在运行的时候，一些“普通的”注意信息，显示的速度很快，而且这些信息全部看的话，没有实际作用，所以尽量将其关闭。

public=true/false

这个条目决定了服务器是否把自己的 IP 发送到其他服务器去。把这个条目设置为 true 对网络来讲是非常重要的，因为一般你要运行的都是公开服务器。只有设置为 public=true 服务器才能够登录到网络上去。

注意：public 默认是设成 false 的，在设置的时候，记得把它改为 true。

threads=10

这个条目主要用来定义服务器能够同时处理的任务的数目，也就是并发的线程数，这个数的值是 Integer（整数）类型，目前 Lugdunum 中此值为 10 且不可更改。

tableSize=3089 (integer)

这里的 table 指的就是前面讲过的 eD2k 服务器中的数据库，存储的是文件名和客户的数据信息。在默认的 ini 文件里这个值是 3089，Integer 类型的，设为其他素数也没有任何影响。如果不去设定，服务器会自动生成一个素数取代。

注意：这里为什么说要设定一个素数呢？因为在 eMule 服务器中，需要时刻处理来自客户端用户的大量搜索和查询请求，那么在服务一端必须要有一个快速的搜索排序机制，设置素数就是为了更快速地搜索和查询。关于此技术背景，请读者自行查阅相关资料。

maxClients=100000

maxClients，从字面意思看，就是最大的客户端数目，这个条目的意思，就是规定了多少客户可以同时连接到服务器的数目。如果 maxClient（最大客户数目）达到了峰值时，也就是此处设定的值 100000，那么，当有客户端在尝试连接到该服务器的时候会出现“Can′t connect to...（无法连接到……）”这样的信息。

注意：关于 maxClients 的最佳值没有统一的规定，根据主机性能、网络条件等不尽相同，需要试试才能知道你的服务器的最佳值，Lugdunum 支持运行中更改可容纳客户端数目，更改后也无须重新启动程序。

type=key/substring

这里的 type 是对系统运行模式的一个设定，type 默认被设置成 key，也就是快速模式；而 substring 模式则是老式的方法，速度很慢。

console=true/false

console 在计算机用语中，就是控制台的意思，设定这个条目的值主要用于决定服务器是否工作在控制台（命令行提示符）状态下。当服务器自动开始运行并且你不想使用“屏幕”命令的时候，就可以设置为 console=false。但这时就不能用键盘给出任何命令了，并且不能显示任何东西。

如果 console=true，就必须在控制台下运行服务器，其实这两样都不是必需的，可根据管理服务器的需要自行设置。Console 默认值是设置成 true。

minVersion=57

minVersion，这个条目是针对与版本相关的设置，minVersion=57 的意思是低于 v.57 的版本不能连接到服务器。当有客户端软件的更新版本出现时，更改这个值是有用的，但请注意当时的形势。比如说，由于一个安全方面的 bug，新版本 v.61 没法支持 ed2k-links，因此没有很多用户升级到它，还有 linux 的版本可能比 Windows 版本老很多等。

如果没有设置 minVersion 值，也就是保留此条目值为空，那么所有版本的客户端都会被服务器接受。

maxVersion=9999

maxVersion 与 minVersion 的意思相对应，这定义了服务器可接受的最大版本号。测试版的客户端程序往往有一个很高的值（比如 1060）——因此这个值应该留为空白，否则得常常更新这个值了。

如果 maxVersion 没有被设置，任何高于 minVersion 的版本均会被接受。

welcome[0]=Welcome to eMuleServer_Test

welcome[1]=share your Files and your upload Bandwith

这是欢迎信息，在登录到服务器时被显示出来。设定时要注意，方括号[]里面的数字是变化的，也就是从 welcome[0]、welcome[1]、welcome[2]……一直写下去，每行都不一样。

注意：虽说可以一直写下去，但欢迎信息不应该太长，否则它们将需要许多带宽——每个字母 1 字节！当出现 1000 客户时，平均每分钟会有 5 ~ 50 个连接。

根据以上的讲解，一个完整的可以使用的 donkey.ini 文件就配置好了，配置过的 donkey.ini 文件内容如下。

```
[server]
name= eMuleServer_Test
desc= http://www.p2p.book
thisIP=123.115.215.7
port=4668
seedIP=123.115.215.8
```

```
seedPort=4668
verbose=false
public=true
threads=10
type=key
tableSize=3089
maxClients=100000
minVersion=
maxVersion=
logFile=false
welcome[0]=欢迎来到 eMuleServer_Test
welcome[1]=**************************
welcome[2]=这只是一个测试哦！
```

3．一些其他的设置选项

在 Lugdunum 中，还有其他的一些设置选项，这些选项也是架设、管理 eD2k 服务器所必须的，有必要掌握好。以下是一些关键配置信息的简要说明。

- LOWIDenable (integer)：如果为 1，LOWID 用户可以登录。默认值为 1。
- LOWIDpercent (integer)：最大的 LOWID 用户比率。建议不要超过 33%，默认值为 20。
- autoservlist (pathname)：如果设置了，服务器会每 225s 将已知的其他服务器列表写入 server。默认值为 none。
- auxportslist (list of ports values)：辅助监听端口列表，16.45 版的新特性。

```
例如：auxportslist=80，443，25，21
```

- blacktime (integer)：黑名单时限。即将客户端 IP 列入黑名单保留的时限。默认值为 3600。
- bverbose (boolean)：如果为真，eserver 会记录下黑名单 IP。默认值为 false。
- connIP (IP address)：当服务器有多 IP 时，指定辅助监听的 IP。和 ftpd 的 virtual host 不同，这里还有防止 Hash Stealers 的功能。
- filter[ ] (filter expression)：滤镜，防止共享某些不合法或不完整文件。

```
例子：
filter[0]=(.part.met)
filter[1]=(.part.stats)
filter[2]=(#FORMAT met)
filter[3]=(#FORMAT part)
filter[4]=(#FORMAT dll)|(#FORMAT sys)
```

- hardLimit (integer)：共享文件数目硬性限制，为避免某些用户共享过度的文件数浪费带宽而设置，拥有超过此数目共享文件的用户将被移出服务器，默认值为 4000。
- login_timeout (integer)：登录时限，在时限内检验客户端获取 HighID 或 LOWID。默认值为 20。
- max_clients_per_ip (integer)：限制同一 IP 连出的客户端数量，默认值为 12。
- 可以防止某些蠕虫/病毒/机器人发起太多连接以填满服务器的资源，但是容易使国内一些宽带的用户进入黑名单。
- maxSearchCount (integer)：从以连接客户中搜索返回结果最大数，默认值为 200。

- maxUDPSearchCount (integer): 从非连接客户中搜索返回结果最大数，默认值为 20。
- maxservers (integer)：服务器被加入服务器 list 的最大值，避免拒绝服务攻击。默认值为 4096。
- maxstrangers (integer)：最大陌生用户的数目，默认为 1000000。
- minEVersion (integer)：可登录服务器的 Emule 最小版本，默认值为 0×26。

注：EMULE 的版本数字为十六进制，范围从 00 到 FF。

- minkeylength (integer)：搜索时关键字的最小长度，默认为 3。
- nbuserIP (IP address)：如果使用了 nbuser 来监听，在这里设置监听机器的 IP，默认为 127.0.0.1。
- nbuserport (integer)：nbuser 监听的端口，默认为 5656。
- ncpus (integer)：设定主机可用的 CPU 的数目。
- nickcommunity (string)：非陌生客户认证的标志，也就是登录服务器需要的 TAG，比如 POPGO 服务器需要的 EDTOON，默认值为空。
- noudpslowsearches (boolean)：拒绝复杂搜索，即拒绝关键字搜索，默认值为 false。
- ping_delay (integer)：ping 延迟时间，服务器会在一定的间隔获取用户总数和每一个用户共享的文件，这个过程叫做 ping。默认值为 400。
- softLimit (integer)：共享文件数目软性限制，为避免某些用户共享过度的文件数浪费带宽而设置，用户超过此数目的共享文件将被服务器忽略，默认值为 1000。
- tcpthreads (integer)：用于接受客户端请求的 TCP 请求的线程数目，默认值为当前主机的可用 CPU 数目。
- trackbademule (integer)：拒绝虚假版本 EMULE 的登录，要与 minEVersion 配合使用，默认值为 30。
- trackemule (integer)：此项如果被激活（设为 1），服务器程序将跟踪 EMULE 的版本，默认值为 1。
- udpsearchers (integer)：为 UDP 搜索动作准备的线程数目，在单 CPU 机器上请设为 1，多 CPU 机器上请设为 2。
- warnfakes (integer)：恶意文件提示，当用户持有或正在下载 fakes.txt 中已知的虚假或恶意文件时，服务器发给该用户的警告信息的数目，默认值为 0（不发送）。

4．常用的服务器命令

- Vc：查看当前服务器中用户登录情况。
- Vs：查看种子服务器的运行情况。
- Vo：查看当前服务器的一些选项的值，比如 IP，端口，软硬限制等。
- name=valve：更改选项的值，比如键入 maxClients＝30000 就是将最大客户端数目设为 30000。
- print name：显示该选项的当前值，比如输入 print maxClients，服务器就会显示 maxClients＝30000。
- g | stats：显示服务器当前的用户情况，搜索状况，端口信息，连接情况。
- Wel：显示服务器的欢迎信息。

- Filters：设置服务器中的文件名过滤。
- Slab：显示当前的内存使用情况。
- Debug：显示服务器的调试信息。
- Reload：重新载入配置文件。
- m message：向客户端广播信息，message 指代广播内容。

7.5　如何“骑”好你的“驴”

以上几节对 eMule 的理论知识进行了详细的讲解，相信读者对 eMule 有了一个更高、更全面的认识。eMule 的成功不仅在于它的技术特点上，更在于它的实际应用中。eMule 作为一款纯 P2P 的文件共享系统，在互联网中有着巨大的应用，本节就从应用的角度来讲一下，如何使用、如何用好 eMule。

7.5.1　使用电驴下载的几点优势

使用 eMule，首先要明白，使用 eMule 到底有什么好，它与其他的下载软件相比又有什么独特的地方。本章 7.1 节中，已经将 eMule 与其他 P2P 软件进行了对比，主要是从技术的角度来进行说明的。下面就从应用的角度来说一下使用 eMule 的好处。

- 使用 eMule 不需要服务器来存放共享文件，节省了服务器架设、海量硬盘、网络带宽。
- eMule 系统中，每个用户端结点都同时是文件下载者和提供者。实际上，在你正在下载但还没下载完整个文件时，你已经可以把你已下载的部分共享给别人了。因为 eMule 是同时从很多文件提供者那里下载所需的文件最后再拼成整个文件的。
- 加入 eMule 网络的人越多，下载速度越快，资源越丰富。
- eMule 共享方便，每个 eMule 客户端可以很简单地指定一个共享目录就可以把自己的文件共享给网络中的其他人了。不必再辛苦地上传到服务器上。

7.5.2　使用 eMule 进行文件下载

本节主要带领读者学习一下，如何使用 eMule 下载文件。

（1）下载 eMule：eMule 下载地址 http://www.emule-project.net/home/perl/general.cgi?l=1&rm=download。eMule 的版本很多，不同的版本有不同的特点和功能，推荐使用 VeryCD 版的 eMule。

（2）安装 eMule：VeryCD 版的 eMule 安装很简单。全中文的安装界面，这里就不再重复说明了。

（3）更改目录：eMule 安装以后，默认的目录是 C:\Program Files\eMule，所有在下载后的文件都会移动到目录：C:\Program Files\eMul\incoming 下，针对这个目录用户可以根据 eMule 的设置选项将其改正过来。选择菜单“选项”|“目录”命令，把“下载的文件”和“临时文件”两个目录选择到不是系统盘（一般是 C:\盘）的分区，如 D:\eMule\incoming

和 D:\eMule\temp。下面还有一个共享目录，可以选择想共享的分区、目录或者文件，在前面勾选就可以共享给其他的 eMule 用户了。设置界面如图 7.33 所示。

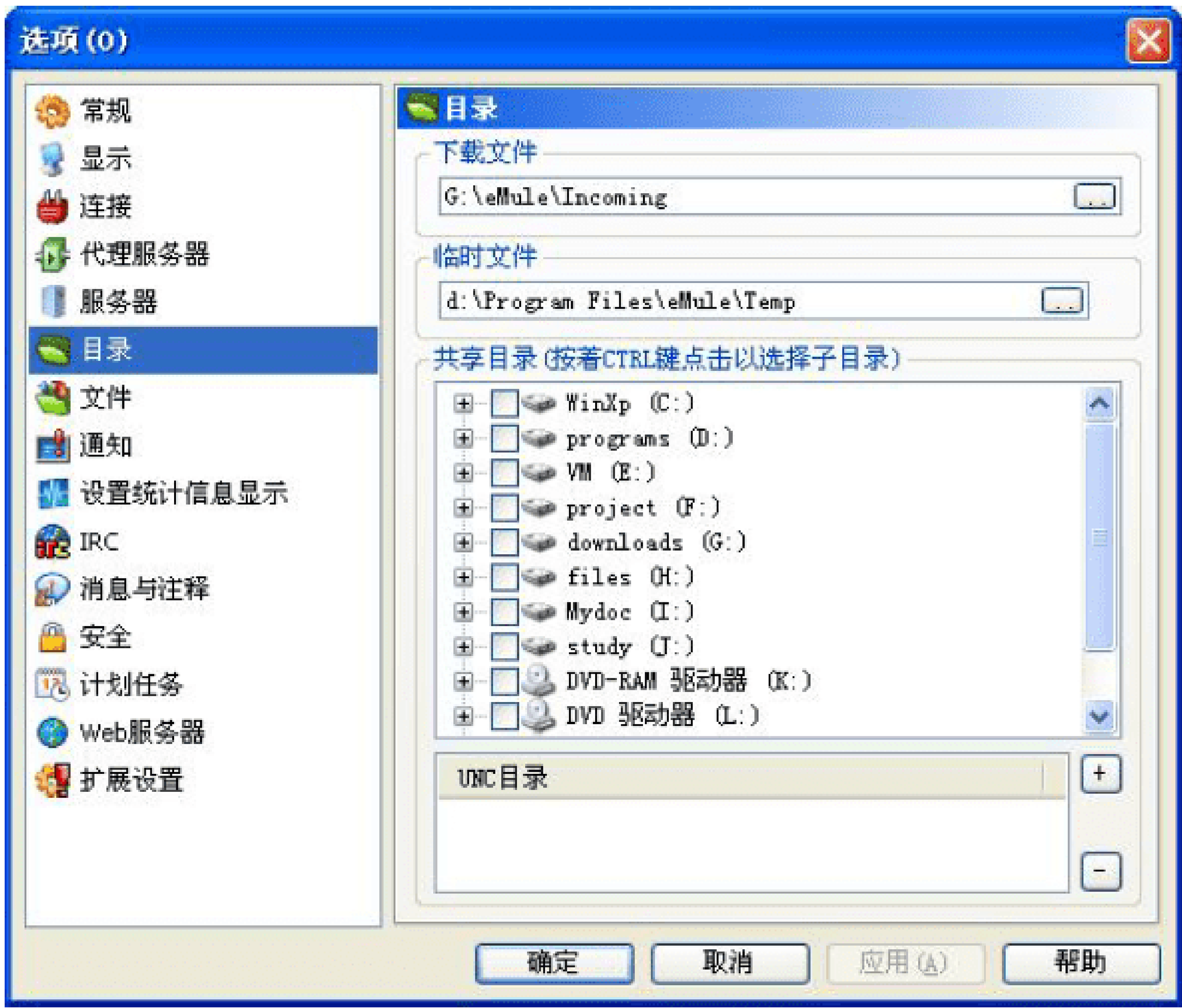


图 7.33　eMule 文件目录和共享目录的设置界面

注意：这个目录的设置也可以在安装 eMule 的时候直接一次性更改。

（4）文件下载：运行 eMule 后，它会自动连接服务器（也可以自己双击连接）。连接成功之后，可以在 eMule 社区或是通过搜索引擎找到 eMule 资源发布的连接，单击这个资源链接 eMule 就会自动将其添加到下载列表当中。如图 7.34 所示，就是 VeryCD 网站上发布的 eMule 资源链接形式。



图 7.34　eMule 资源发布的链接形式

在图 7.34 中还可以看到“复制选中的链接”选项，单击此按钮后，将其粘贴到文本中是这样一个链接形式：

eD2k://|file|Monty.Python.-.%5BMatching.Tie.%26amp%3B.Handkerchief%5D.%E4%B8%93%E8%BE%91.%28MP3%29.rar|49330986|3ee7d070083fdbe576c419c62071f9cc|h=SE7LV6LNTWWCSMK6S7GWAY2UXSRBZBCG|/

下面以就是“Monty.Python.-.[Matching.Tie.&.Handkerchief].专辑.（MP3）.rar”这个文件的链接为例，结合上文对 eD2k 链接的讲解，再解释一下电驴文件 eD2k 链接里面的相关信息，上面展示了此文件 eD2k 链接的完整形式。链接以“|”划分可以分成 3 部分。

- 文件名：虽然最直观醒目，但是最不关键，作用仅是便于搜索，文件名的中文编码格式都经过了转义。
- 文件大小：也没有什么用，主要用来说明文件的大小，对于一些视频文件而言，一般情况是越大越好，理论上讲越大越清晰。
- 文件 ID：又叫做文件 Hash，是区别于其他文件的根本标识，这才是 eD2k 链接里面的关键，在分块下载、检查文件的完整性方面都需要用到文件 Hash。

很多文件即使它们的文件名不一样，但是只要文件 ID 一致，eMule 服务器就视同为同一个文件。如果想知道要下载的文件是否以前已经下载过了，唯一的操作办法就是将每次下载文件的文件 ID 保存到文本文件里面（当然保存 eD2k 链接更简便），然后下载之前查找一下你要下载文件的文件 ID（千万不可查找 eD2k 链接）是否在该文件中即可判定。

单击图 7.34 中的“下载选中的文件”按钮就可以执行下载了。然后就会在 eMule 下载任务中就可以看到这个资源，如图 7.35 所示。

图 7.35 说明了，一个资源已经纳入到 eMule 系统中，正在进行下载。

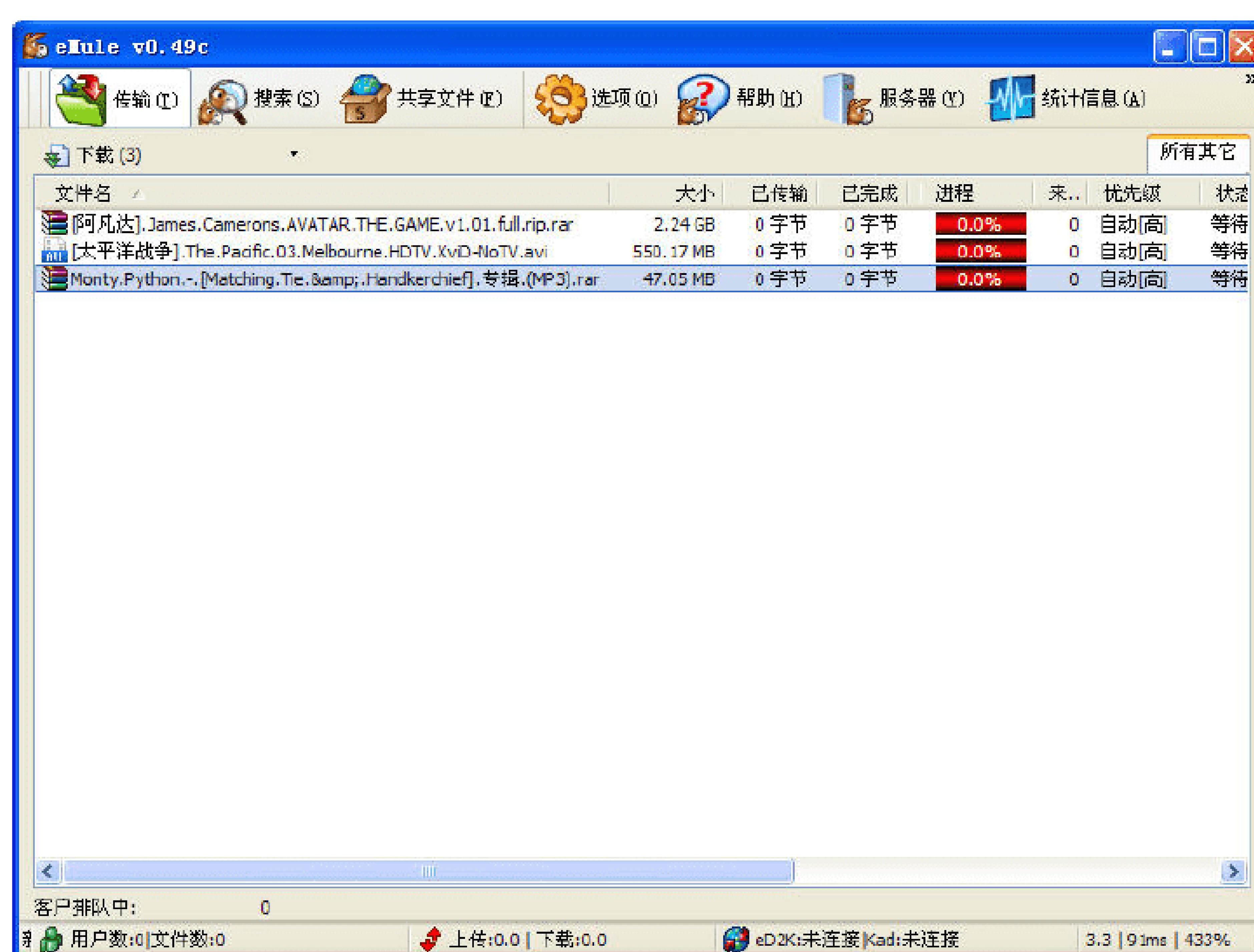


图 7.35　eMule 系统中新加入资源下载的示意图

7.5.3　在 eMule 中如何上载文件

在 eMule 中上载文件分两种情况，一种是在 eMule 下载的同时上载文件（也就是将正在下载的文件分共享给其他 eMule 用户），另一种将本地硬盘上的文件共享出来，进行上载。

1．上载通过eMule下载的文件

在 eMule 下载的时候，系统会自动在下载的同时也开启上传的功能，所有参与共享的文件都会被 eMule 指定到共享目录下、在 eMule 工具栏中单击“共享文件夹”就可以在左

侧展示出共享的目录菜单，eMule 会将两类文件自动共享，一类是已经下载完成的文件，如图 7.36 所示。

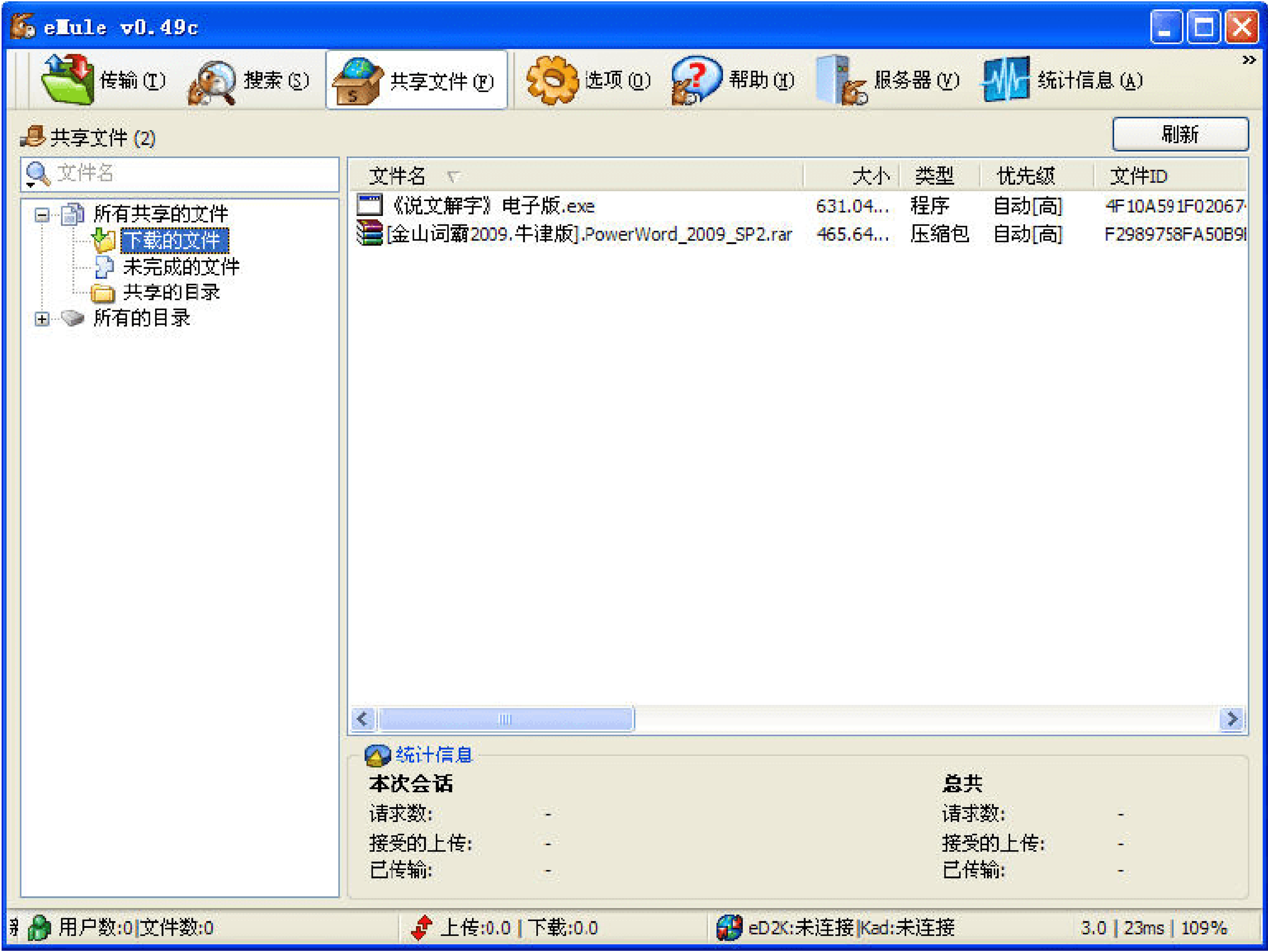


图 7.36　在 eMule 中已经下载完成的文件共享示意图

另一类是正在下载或暂停下载而未下载完成的文件，如图 7.37 所示。

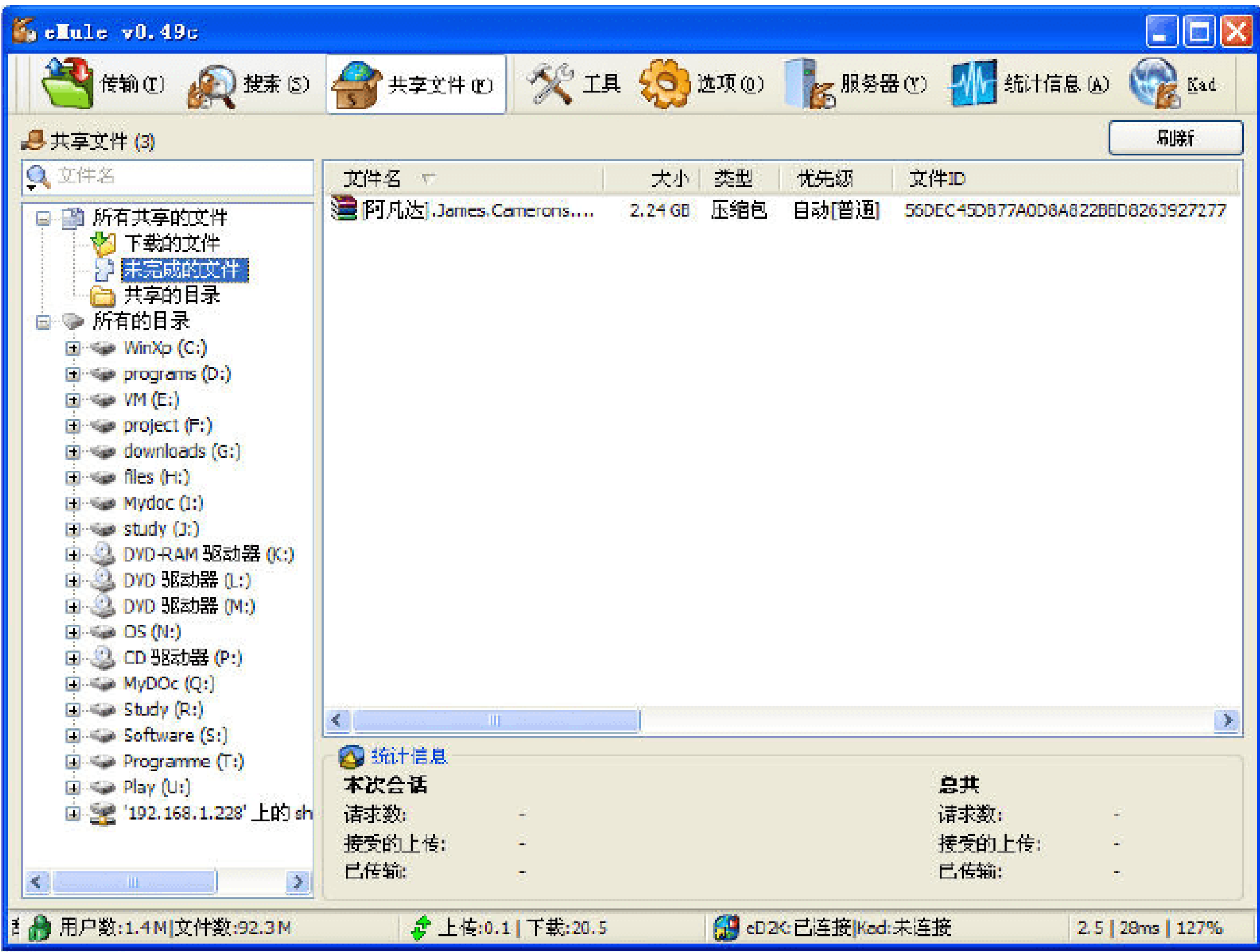


图 7.37　未完成下载的文件共享示意图

针对这些文件，如果想将其 eD2k 链接公布到 Web 网络中，只需打开共享目录，然后找到要公布的文件右击，在弹出的快捷菜单中选择“显示 eD2k 链接”选项再将其复制到剪贴板，然后在 Web 网络中公布即可。

2．上载本地硬盘上的文件

eMule 另一个上载文件的途径就是将本地硬盘上的文件共享出去。在 eMule 系统中，选择工具栏中的“共享文件”选项，然后展开左侧的所有目录（本地硬盘上的目录），找到想要共享的文件，单击“共享此目录”即可，如图 7.38 所示。

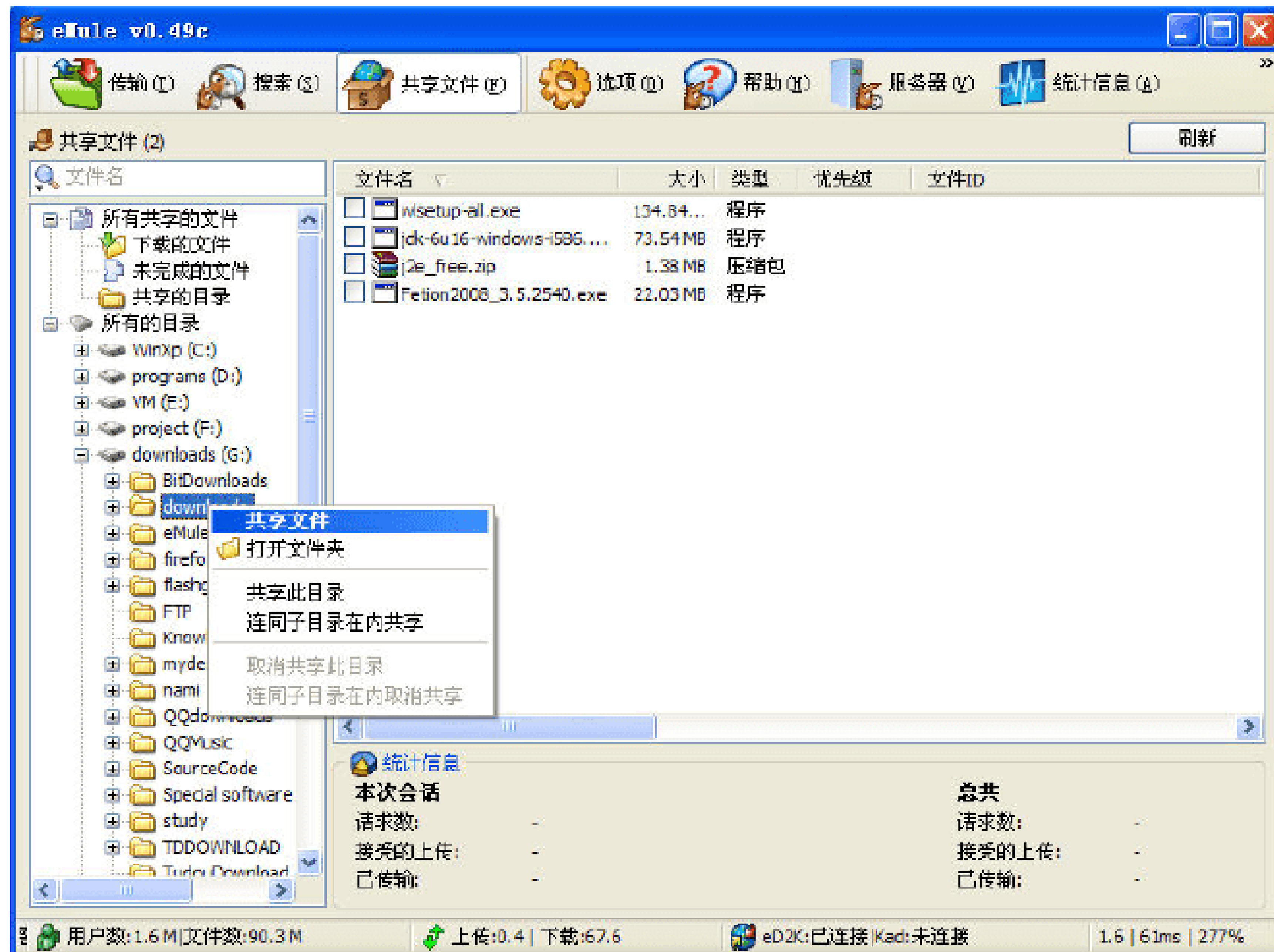


图 7.38　将本地文件共享到 eMule 中的操作示意图

7.5.4　通过 VeryCD 发布 eMule 资源

VeryCD 是中国的基于开放源码 P2P 网络共享软件 eMule（电骡）的媒体资源提供网站。VeryCD 开始于 2003 年 9 月，并使用 eMule 作为基本共享客户端。VeryCD 的目标与使命是通过开放的技术构建全球最庞大、最便捷、最人性化的资源分享网络。它由网站社区操控，由网友提供资源。VeryCD 已经成为国内媒体资源最丰富的网站之一。网站地址为 http://www.verycd.com/。

使用 VeryCD 的发布系统必须拥有 VeryCD 用户账号，目前不开放注册，只能由老会员推荐注册。详情请见 VeryCD 论坛注册页面。

注意：当你的 eMule 使用达到一定等级的时候，就可以注册一个 VeryCD 的用户账号了。

1．发布前的准备

将准备发布的文件准备好。首先，需要在 eMule 里面添加共享目录，默认目录为 C:/Program Files/eMule/incoming，可以把要发布资源的所在文件夹添加为共享。

设置步骤：在选项对话框中选择“目录”选项。在“目录”选择页里的共享文件夹前面打勾，单击“确定”按钮，如图 7.39 所示。

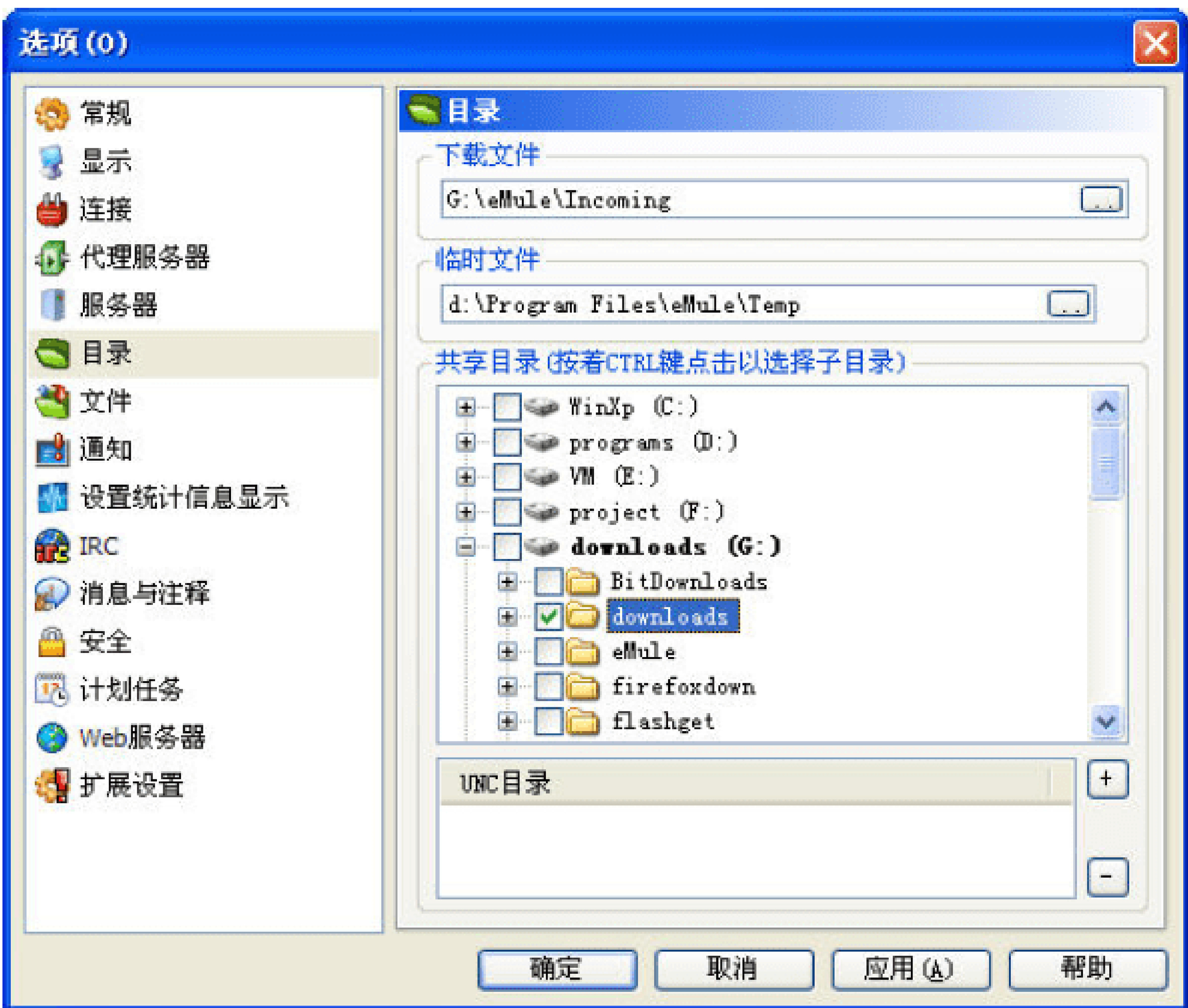


图 7.39　eMule 中设置共享目录的操作界面

注意：设置共享目录的方法有很多，也可以按上面所讲的共享本地硬盘文件时的设置方法。

添加好共享文件之后，在 eMule 的共享菜单下可以看到已经添加的共享文件。（没看见的话多刷新几次）。然后，右击要发布的文件，在弹出的快捷菜单中，选择“优先级”|“发布”命令，如图 7.40 所示。

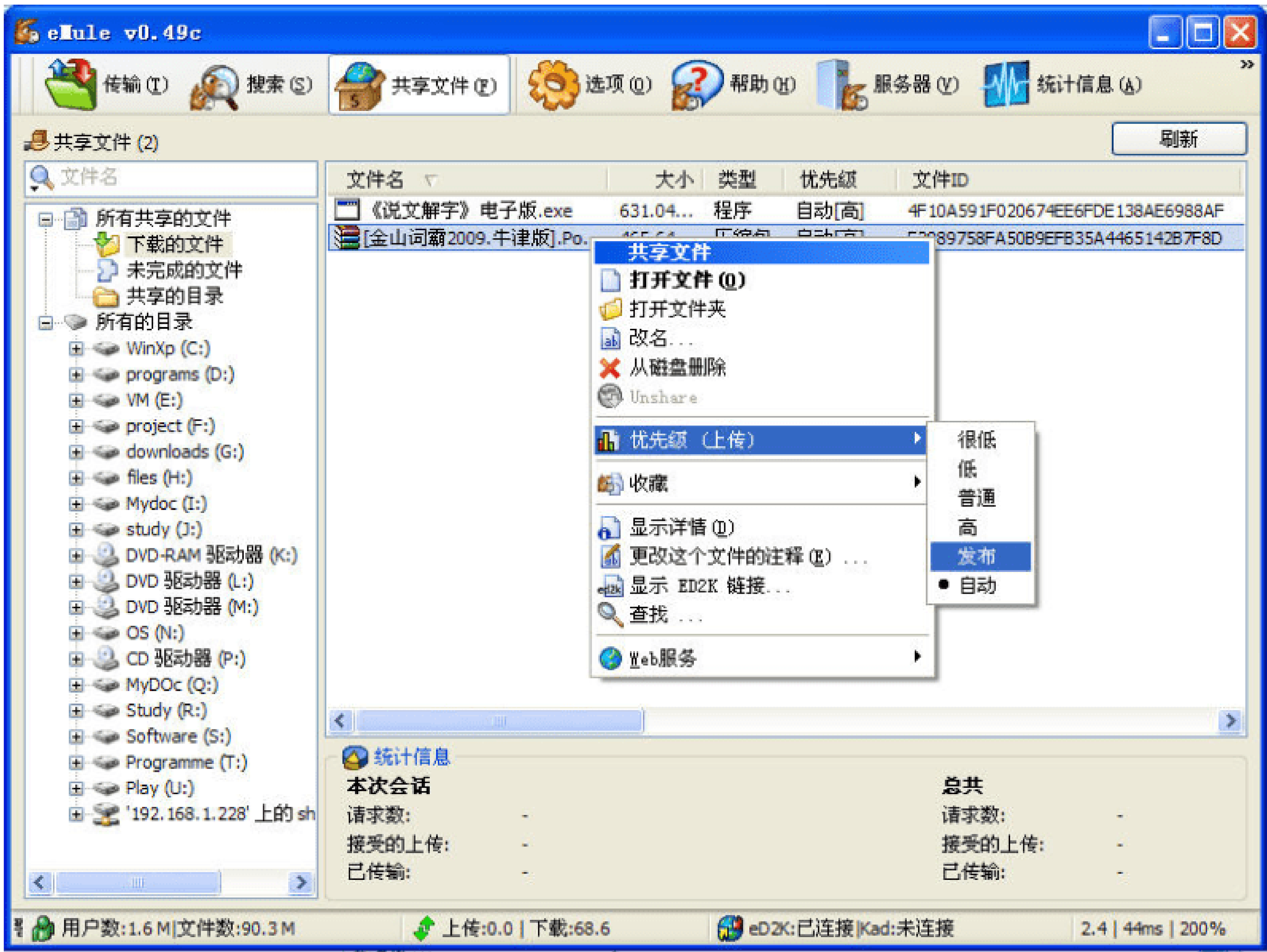


图 7.40　将共享文件进行发布的设置界面

将共享文件设置成发布以后，还要找到共享文件的 eD2k 链接，将此链接复制下来，保存到文本文件中。查看共享文件的 eD2k 链接的方法是：右击共享文件，在弹出的快捷菜单中选择“显示 ED2k 链接”|“复制”命令，如图 7.41 所示。

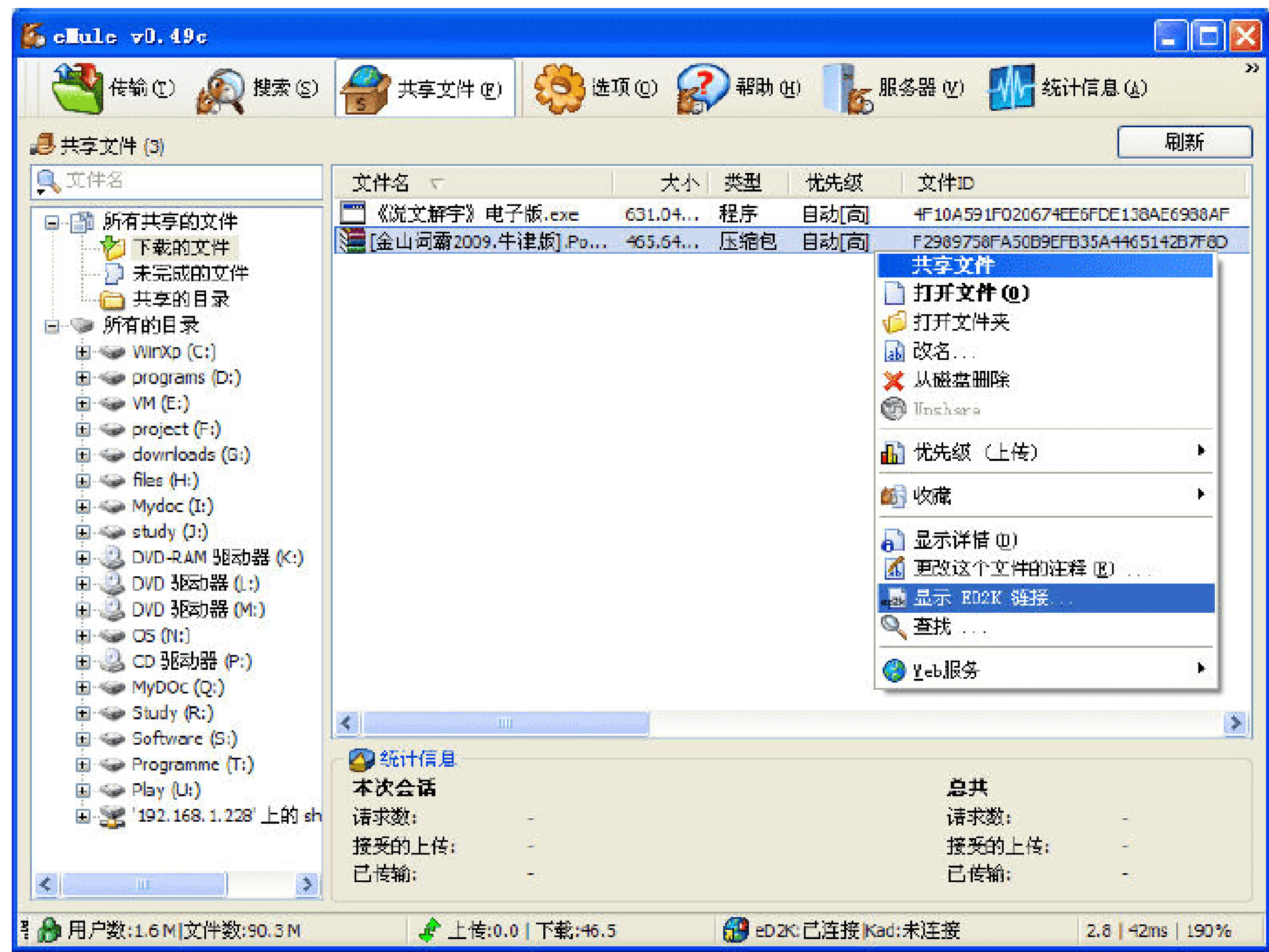


图 7.41　查看共享文件 eD2k 链接的界面

按图 7.41 所示的操作，显示出 ED2k 链接后，将链接内容复制下来。

2．在VeryCD上发布共享资源

到 www.VeryCD.com 网站上，单击提交资源，填写发布资源的详细介绍、资源信息、图片引用的 URL 等。填写完成后，将之前复制好的 ED2k 链接粘贴进去，就可以发布这个资源了。

注意：如果是链接网络上的图片，直接右击图片，查看属性就可以找到此图片的链接 URL。

步骤：填写资源分类→中文名称→其他必要信息→资源内容介绍→粘贴 ED2k 链接。操作界面如图 7.42 所示。

分类：	电影　爱情 *
资源格式：	DVDRip *
电影中文名：	*请在这里输入电影中文名。
电影英文名：	请在这里输入电影英文名。
电影别名：	请在这里输入电影的其他名字，
版本说明：	如发布资源为特殊版本(未分级
发行时间：	请在这里输入电影发行时间(YY 可(YYYY)

图 7.42　在 VeryCD 中发布共享资源的操作界面

7.5.5　如何使用 eMule 进行文件搜索

通过 eMule 下载想要的文件，需要根据想要的关键字进行文件搜索。文件搜索是用好 eMule 的前提，eMule 中有大量的非常有价值的资源，但如果不能快速地找到有用的资源信息，也就无法有效使用 eMule 了。

1．利用搜索引擎在Web中搜索eMule资源

搜索 eMule 资源有两种方式，其中之一就是利用通用的搜索引擎来搜索 Web 网络中发布的 eMule 资源，如图 7.43 所示，就是 VeryCD 中搜索 eMule 资源的方法。



图 7.43　利用搜索引擎搜索 eMule 资源的方法

在图 7.43 中，只要输入有搜索的关键字，大部分情况下也能找到理想的资源，也可以利用 Google、Baidu 等搜索引擎在 Internet 上搜索 eMule 资源。搜索到资源后，想办法找到发布此资源的 eD2k 链接，就能执行下载了。

2．从eMule客户端中搜索eMule服务器的资源

另一种搜索方式，就是利用 eMule 客户端的搜索功能，直接到 eMule 服务器上搜索文件，这种搜索方式也是 eMule 真正意义上的搜索。具体的搜索步骤如下。

（1）首先要与 eMule 服务器取得连接。在 eMule 工具栏中，选择“服务器”选项，会弹出一个服务器列表的界面。选择其中一个服务器点右击，在弹出的快捷菜单中选择“连接到所选的服务器”（或者是双击所选的服务器）选项，这样，eMule 客户端就会发起一个与 eMule 服务器的连接。操作界面如图 7.44 所示。

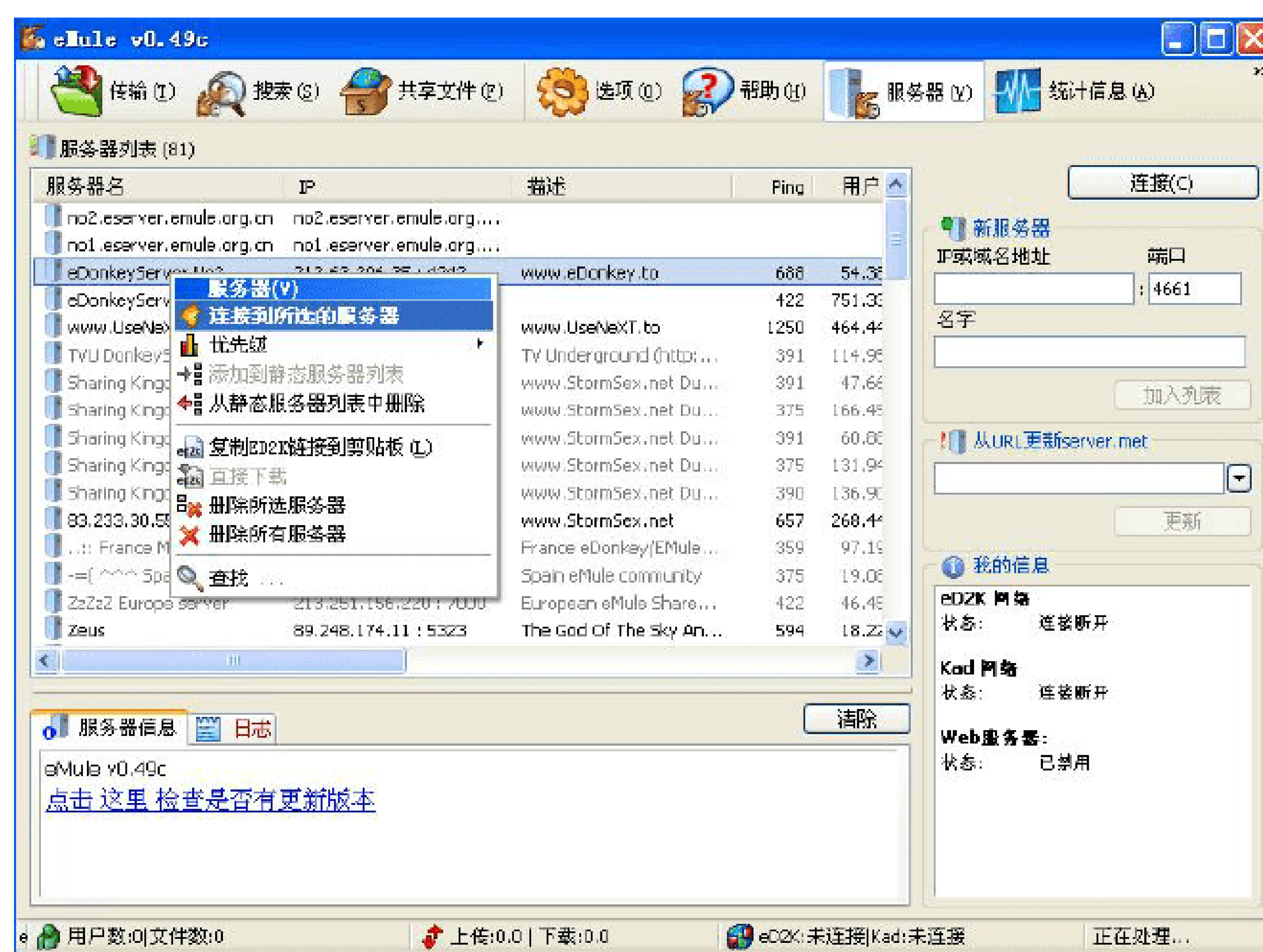


图 7.44　eMule 客户端与服务器连接示意图

🔔注意：在链接服务器的时候，有时候会出现连接不成功、拒绝连接的情况，（这在 eMule 协议分析里也讲过）出现这种情况的时候，只需重新选择一个服务器，重新连接一次即可。

客户端与服务器连接成功后，会在 eMule 的“日志”文件里显示出此信息，如图 7.45 所示。



图 7.45　eMule 服务器连接的日志信息

图 7.45 所示的就是与服务器连接成功的日志信息，新的服务器也为 eMule 客户端分配了一个新的客户 ID。

🔔注意：以上的操作过程，要和 eMule 协议规范结合起来学习，理解 eMule 客户端与服务器通信过程。

（2）确定 eMule 客户端与服务器连接成功后，就可以搜索文件了。具体操作步骤如下。

在 eMule 系统的工具栏中，单击“搜索”菜单，就会出现一个搜索界面，如图 7.46 所示。在这个搜索界面中有很多搜索选项可以选择，用户可根据自己的需要自行设定搜索参数。

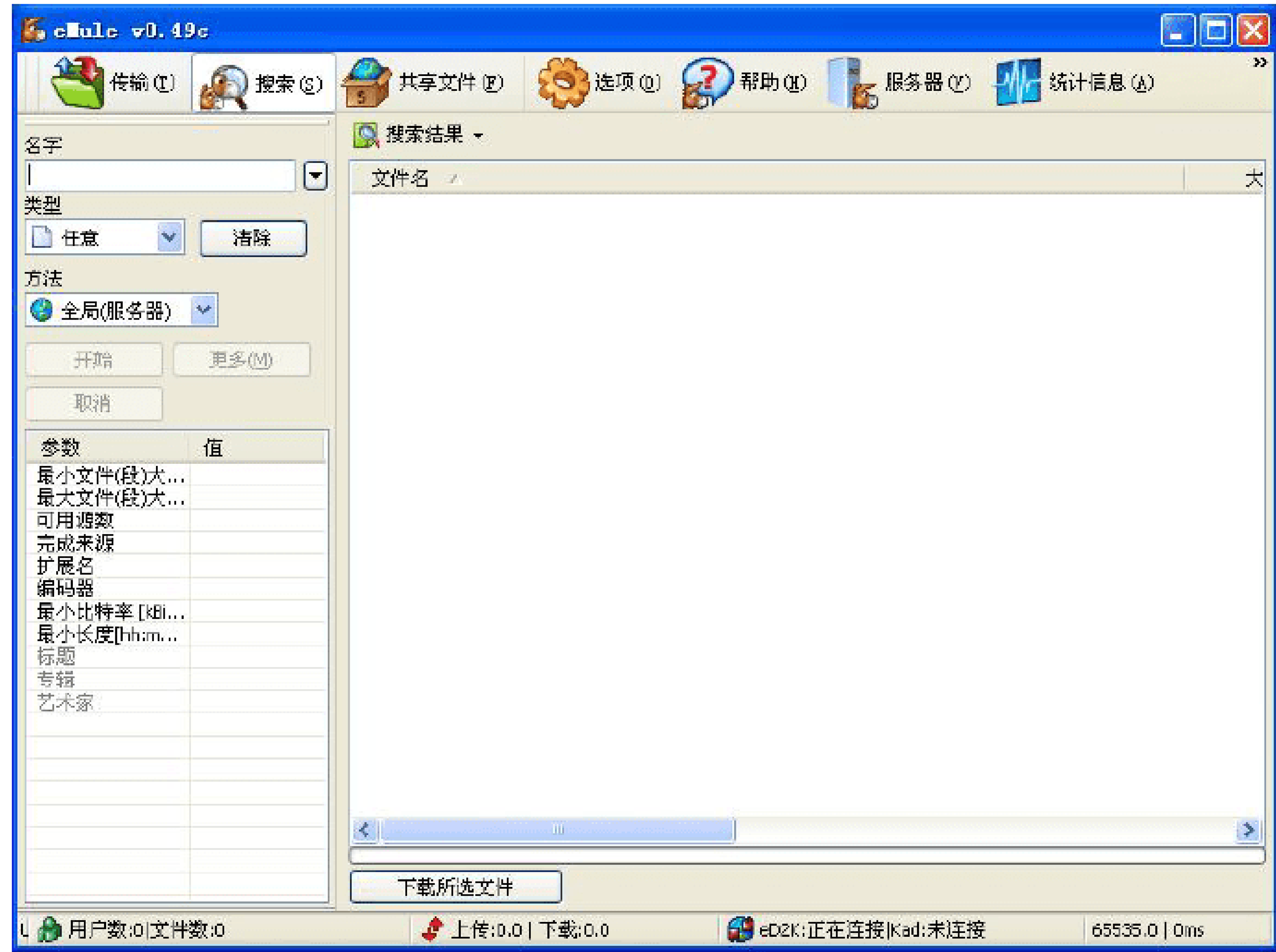


图 7.46　eMule 系统中搜索功能界面

在“名字”文本框里输入关键字，“类型”可以选择任意（推荐方式）或者视频（无法搜索 dat 文件），这里选择“任意”。在“方法”选项里默认的是“自动”，本次搜索就选择默认的设置。单击“开始”按钮，eMule 客户端就开始向服务器发送搜索信息了。

稍等片刻，eMule 服务器就会把搜索结果返回给客户端，这时就会发现在 eMule 搜索结果列表中列出了很多可下载的符合关键字特征的文件。如图 7.47 所示，就是对 eMule 关键字执行搜索的结果。

图 7.47 显示的是，在 eMule 网络中，搜索到了 153 条与 eMule 关键字有关的文件。当然，如果选择的服务器不同，搜索结果也会不同。

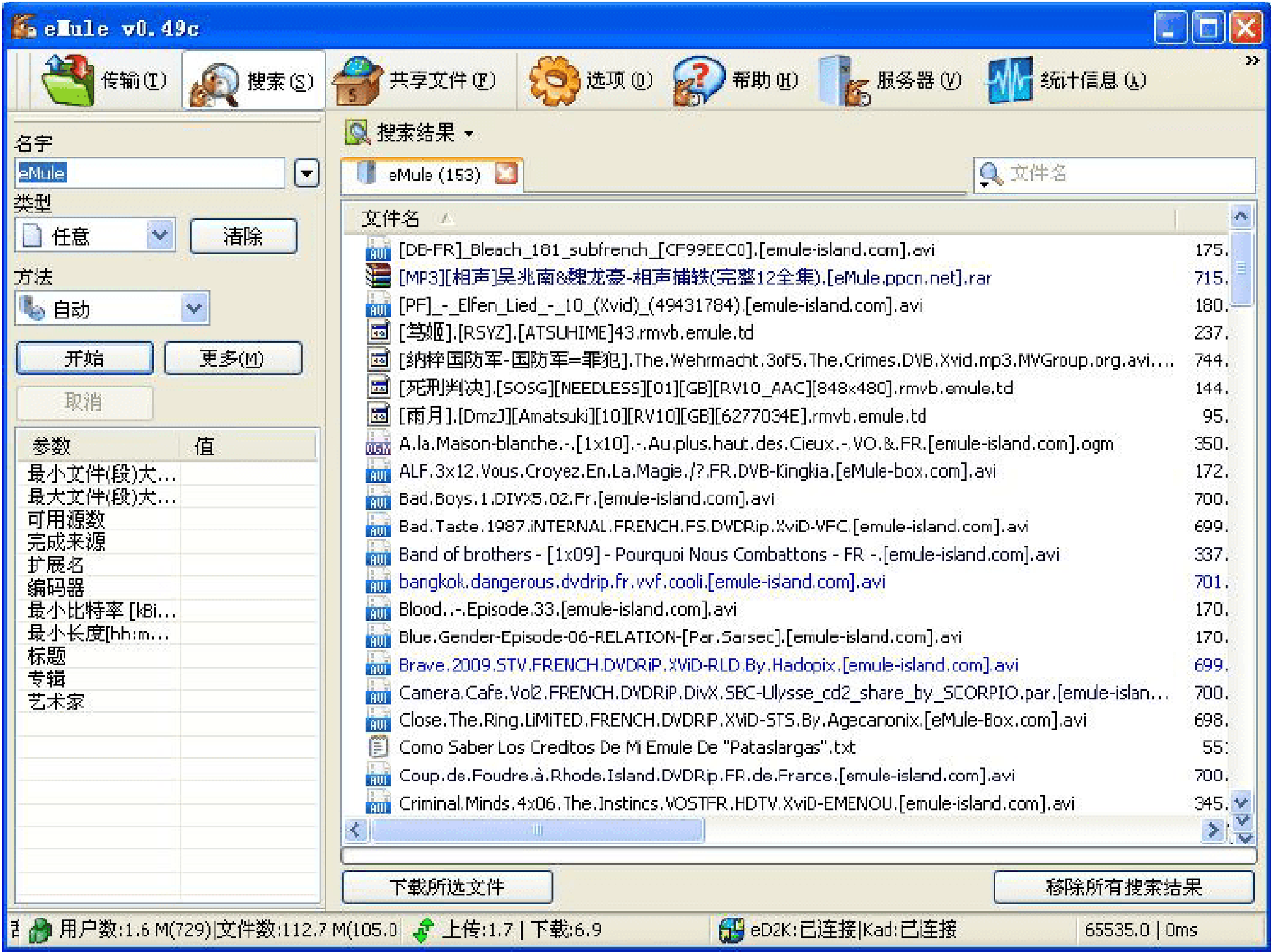


图 7.47　eMule 执行搜索的结果界面

在搜索的结果列表中，选择“来源”多的文件，双击就可以下载了。如果要保存搜索的文件信息，可以在搜索结果窗口里，同时按下 Ctrl+A 键全选，然后右击。在弹出的快捷菜单中选择“复制 ED2k 链接到剪贴板”，或是“复制 ED2k 链接（HTML）到剪贴板”选项，最后剪贴到一个文件中保存即可。如果是复制 HTML 格式的 ED2k 链接，可直接另存为 HTML 网页进行发布，如图 7.48 所示。

有了 eMule 搜索的功能，理论上讲可以搜索到所有 eMule 服务器上的文件，在执行 eMule 搜索的时候，还有几点需要注意。

（1）当向 eMule 系统提交共享文件时，这些文件最好是能提供一些可查询的特征字，这样，当其他的 eMule 用户在查找文件的时候，只要输入共享文件所包含的任何一个字符或者字段就可以搜索到了。

（2）信息搜索的时候应全面，如分别用简体中文、繁体中文、英文查找，这样总可以找到你所需要的文件。如果要查找某个演员出演的片子，最好也是用简体中文、繁体中文、英文分别检索，演员的英文名可以用 Google 检索查询。若想检索到更多的文件信息，建议最好用英文查找，毕竟用电驴的还是以西方人居多。

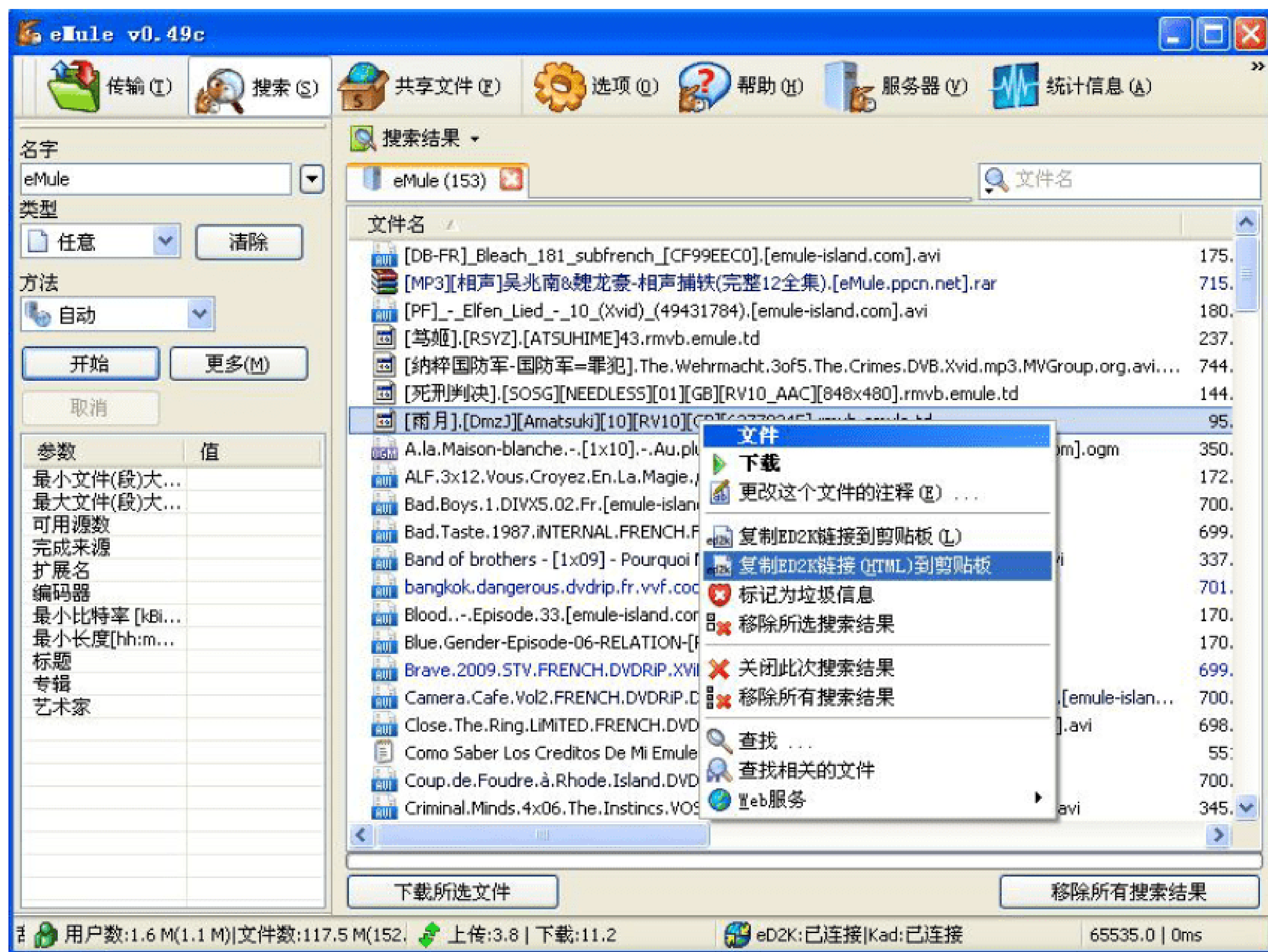


图 7.48　将搜索结果进行发布的操作界面

（3）选择“来源”数目多的文件下载，这样不会因为提供来源的某一人关机而使你无法下载这个文件了。

eMule，是多点、对等下载的文件共享系统，不存在续传的问题，也就是说，登录服务器中的任何一个人一旦共享某一文件，那么 eMule 系统可以保证能完全的下载此文件。

7.5.6　eMule 的主菜单及简要说明

eMule 的菜单非常直观，打开 eMule 客户端后，在 eMule 顶部的工具栏中展示的就是 eMule 菜单，如图 7.49 所示的就是 eMule 的菜单项。



图 7.49　eMule 系统的菜单界面

用户可以根据自己的喜好来定制工具条，从中增加或删减菜单项。

注意：图 7.49 所示的是 eMule V0.49C 版本的菜单情况，不同的版本可能有所不同，读者注意区分。

下面，分别简要介绍一下这些菜单项的含义。

（1）传输：显示的是当前 eMule 在下载文件过程中的传输状态，可自行设定表头信息，以查看每个文件完整的下载状态。也可以右击某一个文件，在弹出的菜单中可以查看此文件详细的下载信息。

（2）搜索：就是在 eMule 客户端进行文件搜索，上文已有讲述。

（3）共享文件：可以查看共享的文件，连正在下载的文件也计算在内。选中文件右击可以更改文件注释。

（4）选项：关于 eMule 各种选项的设置和相关参数的配置。

（5）帮助：关于 eMule 的一些在线帮助信息，会自动链接到 http://www.eMule-project.net/ 网站，展示相关帮助信息。

（6）服务器：以列表的形式，展示当前 eMule 网络中的服务器信息。

（7）统计信息：指 eMule 在运行过程中，关于下载、上传、客户、服务器、时间等各类信息数据的统计情况，对研究 eMule 的网络行为和特征、全局性观测 eMule 的状态有重要作用。

（8）IRC：eMule 附属的一种网上聊天。

注意： IRC 聊天是网上聊天的一种方式，它是 INTERNET RELAY CHAT 的缩写，意思是因特网继传聊天，通过特殊的协议（IRC 协议），大家连到一台或者多台 IRC 服务器上进行聊天。它的特点是速度快（几秒钟内你就可以看到对方的“讲话”），功能多，所以通过 IRC 聊天是全世界网友的最佳选择。因没有经过实际验证，eMule 中的 IRC 是否具备以上功能还不得而知。

（9）消息：一种即时通信系统，类似于 QQ、MSN 等，可以和其他客户端的用户进行在线聊天。

（10）断开/连接：控制服务器的连接状态，可以断开或连接服务器。

（11）Kad：关于 Kad 连接的一些信息，以列表的形式展示。

（12）工具：和 eMule 相关的一些工具，单击“工具”按钮后，其下拉菜单如图 7.50 所示。

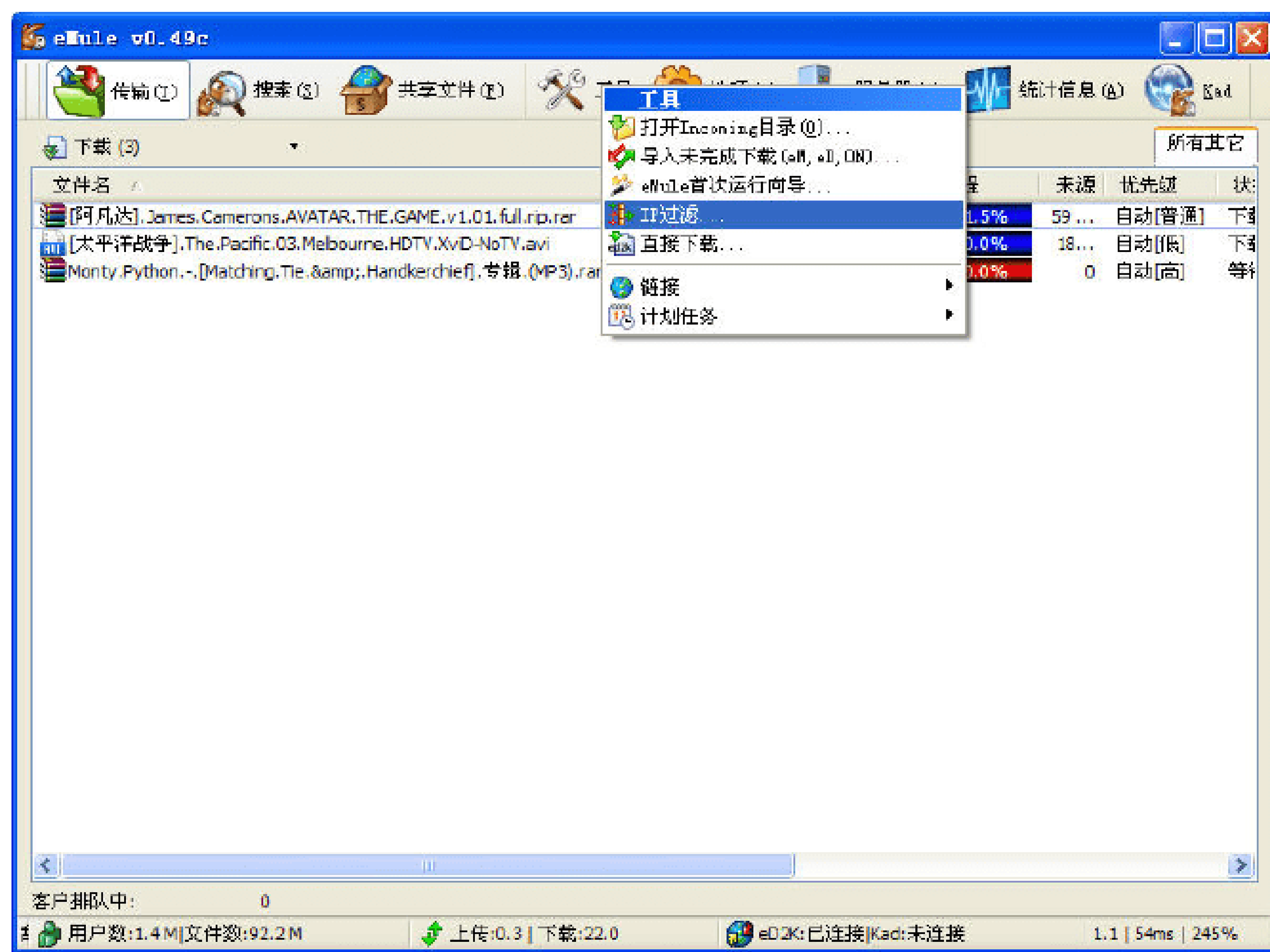


图 7.50　工具菜单的功能显示界面

以上这些菜单项，只要打开研究一下，或是查阅一些在线文档，就可以很轻松地学会相关操作，这里不再赘述。

7.5.7　如何使 eMule 下载加速

几乎每个使用 eMule 的用户，在下载文件的时候都希望自己的“驴”尽可能地快，而 eMule 的下载速度是由多方面的因素影响的。在使用 eMule 的过程中，可以通过一些简单的设置来提高 eMule 的下载速度。

1．给eMule下载分配足够大的空间

在用 eMule 下载的时候，一般都是大文件，所以要预先给 eMule 的下载目录分配足够大的空间，这样可以避免下载的文件占满整个系统分区。宽裕的空间也可让 eMule 进行更快的读写操作。

单击工具栏中“选项”按钮，在选项对话框在侧单击选择“目录”，再在右侧将默认的下载目录和临时文件目录移到其他空闲的分区即可。

2．选择优质的eMule服务器

eMule 下载速度的快慢，与其连接上的服务器有直接关系，如服务器上索引的共享资源的数量、连接上的 eMule 客户端等。选择一个好的、优质的下载服务器是 eMule 提速的一个重要条件。

较简单的方式是使用广大“驴友”提供的服务器列表，最新的安全 eMule 服务器列表可以从网站 http://eMule-fans.com/上获取，也可以通过搜索引擎快速找到，例如 Google，在 Google 的搜索框中输入“server filetype:met”（实际搜索中没有引号），回车后即可看到很多.met 类型的 eMule 服务器文件。将得到的 server.met 文件添加到 eMule 所在目录的 config 目录下，覆盖同名文件即可。

注意：在网站 http://eMule-fans.com/，通过页面链接可以找到最新的 eMule 服务器列表。在这个列表中，直接单击 ED2k 链接，就可以将最新的服务器加入到 eMule 服务器列表中。

3．尽量取得HighID

已经讲过，在 eMule 中有 HighID 和 LowID 之分，HighID 因为能共享到更多的资源，因此下载速度会快很多。所以要想 eMule 有更快的速度，就要想办法让自己的 eMule 客户端获得 HighID。关于 HighID 的获取方法，在本书的第 5 章已有相关说明，请参阅相关内容。

4．通过修改版的eMule提高速度

上文提到过，因为 eMule 是开源免费软件，所以有很多针对 eMule 的一些修改版本。其中就有主要针对 HighID 和 LowID 的用户的两个版本，使用这两个版本的 eMule 可以适当地降低客户 ID 影响，而在一定程度上提高速度。

注意：这种方法只是根据 eMule 版本中的功能描述得出的，并没有经过实际测试和有效数据的验证。

针对 HightID 的用户，可以使用 ScarAngel 修改版，ScarAngel Mod 是基于 Xtreme 的二次 Mod，继承了 Xtreme 的一切特性，并且添加了一些实用的功能。在 http://eMule-fans.com/eMule-49b-scarangel-31/中可以下载，在安装包内有详细的使用说明，读者一看就可以明白。

在网址 http://work.newhua.com/cfan/200823/eMule_0.48a_VeryCD_v080917green.rar 处下载 VeryCD 的绿色修改版及相关的配置文件，可以满足 LowID 的用户的需求，具体设置方法参看压缩包内的详细使用说明即可。

注意：以上提供的网址，可能由于时间关系，有的失效或是版本不一致，读者只需根据一些关键字说明在搜索引擎中重新搜索即可。

5．踹开吸血驴

吸血驴，主要指那些只下载，不上传的 eMule 客户端，或者上传的很少、只上传正在下载的资源的客户端等，其他的有着类似行为的 eMule 客户端也可以叫吸血驴。这种行为造成的最明显的危害就是众多资源下载的速度变慢，资源容易产生断档等。完全违背了 P2P 的共享精神，为了有效提高自身的下载速度，要毫不犹豫地踹开那些吸血驴。

在 eMule 社区中，反吸血驴（DIP，Dynamic leecher Protection，动态返吸血驴防护）插件，专门用来对付这类败“驴”。

首先，从 http://eMule-fans.com/category/news/plugin/dlp/处，下载“反吸血驴”插件，下载并替换 eMule 所在目录下、config 目录下的同名文件。然后运行 DLP_Updater.exe 程序升级反吸血驴插件，最后再运行电驴，这样即可有效防止吸血驴了。

以上就是几种简单的提高 eMule 下载速度的方法，其实 eMule 本身是一门很大的学问，希望读者在不断的摸索、探究中学到更多的知识。

7.6　本 章 小 结

本章从理论到实践系统地讲解了 eMule 的整个知识体系，eMule 作为 P2P 技术中最经典的应用之一，学习并理解 eMule 对基于 P2P 技术的应用开发有重要的指导作用。作为广受欢迎、成熟的文件共享系统，学习 eMule 对日常的网络生活中大文件的下载、本地文件的分发与共享也有十分重要的意义。

本章从理论的角度简要介绍了 eMule 和基础知识，接着从网络结构的角度分析了 eMule 的工作原理，重点讲解了 Kad 网络的原理。然后带领读者详细的分析了 eMule 协议规范，理解并掌握 eMule 协议是进行 eMule 开发、从事 eMule 研究必不可少的要求。在 eMule 知识进阶里，系统地讲解了 eMule 里一些重要的特性和知识点，以及 eMule 服务器的架设，由以上这些知识完全可以开发出一个 eMule 服务器加客户端的完整的文件共享系统。最后，讲解了 eMule 的使用方法，演示了如何使用 eMule 进行文件下载、搜索资源、上传分发文件、发布 eMule 资源等方法，还提出了一些操作技巧如加速下载、增加积分等。

总之，学完本章，读者应该对 eMule 的系统知识有个全面的掌握，对 P2P 技术的实质有更深的理解，能够根据 eMule 协议规范进行 eMule 客户端的开发。

第 8 章　基于 P2P 的 Skype 即时通信技术

相信每一个使用电脑的用户都非常熟悉 IM（即时聊天）软件，如 UC、MSN 和 QQ 等。互联网的迅猛发展，在很大程度上已经改变了人们的通信方式，具备语音通话、即时消息、文件传输、视频交流等功能的即时通信系统已经成为人们日常网络生活中不可或缺的交流平台。在众多的即时通信软件中，Skype 显得尤为突出。

Skype 是一款基于互联网且使用 P2P 技术的通信软件，它兼顾电话和即时通信两大功能于一身，采用新兴的 P2P 技术。它不仅完全可以实现即时通信的功能，在语音通信方面也可以和目前的电话通信媲美。因此，Skype 在商业应用中有着极大的发展潜力，在基于 P2P 技术的研究应用中也有着巨大的科研价值。本章将重点讲解基于 P2P 的 Skype 即时通信技术，使读者不仅是在技术上，也在应用和开发上对 Skype 的整个知识体系、实现原理、基本应用等方面，有更加深入的理解。

本章要讲解的重要知识点如下。

- 即时通信技术：从概念、历史、背景、原理及基本功能等方面，系统地了解即时通信技术的基本知识。
- 基于 P2P 的 Skype 技术：理解 Skype 在 P2P 技术中的体现，了解 Skype 的 VoIP 功能，掌握 Skype 的功能特点及技术优势。
- Skype 技术：重点理解 Skype 的技术实质，从网络结构、协议特点、通信流程、安全机制等方面掌握 Skype 的整个技术内容。
- Skype 的应用及发展：了解 Skype 的基本使用方法，了解 Skype 在不同领域的应用及 Skype 的发展前景。

8.1　什么是即时通信技术

Skype 属于即时通信的应用范畴，在讲解 Skype 技术之前，将先带领读者了解一下，什么是即时通信技术。本节的重点就是讲解即时通信的相关知识。

8.1.1　即时通信的概念

即时通信也称为 IM，是 Instant Messaging 的缩写，中文翻译成“即时通信”。或“即时通信”的意思。

注意：下文出现的 IM 的英文缩写，指的是即时通信的意思。

根据美国著名的互联网术语在线词典 NetLingo 的解释，即时通信的定义如下：“Instant

Messaging（读成 I-M）缩写为 IM 或 IMing，它是一种使人们能在网上识别在线用户并与他们实时交换消息的技术”，在成熟的即时通信系统被发开发出来以前，很多人称电子邮件是一直以来最酷的在线通信方式。

注意：NetLingo 的主页为 http://www.netlingo.com/，对 instant messaging 的原文解释是 Atechnology that gives users the ability to identify people online and to exchange messages with them in real time。

作为一个即时通信系统（IM 系统），其最典型工作方式是这样的，当好友列表（buddy list）中的某人在任何时候登录上线并试图通过你的计算机联系你时，IM 系统会发一个消息提醒你，然后你能与他建立一个聊天会话并键入消息文字进行交流。这种交流是双向的、即时的。

注意：如果你用过 MSN 或者 QQ 及其他的任何一款聊天软件，你就能想象到它的工作过程。

在早期的即时通信系统中，只能传输文本信息，而现在的即时通信系统，不仅能传输文本信息，还可以传输语音、视频、文件等。IM 被认为比电子邮件、BBS 及在线聊天室等更具有自发性、即时性的在线通信系统。

注意：BBS 的英文全称是 Bulletin Board System，翻译为中文就是“电子公告板”，具有发布信息、在线讨论、聊天等功能。在网络中，BBS 常用来进行信息的发布和传递、用户反馈等，多用于网民之间的互动、交流。

除 NetLingo 的定义之外，还有一些其他定义，比如，定义即时通信为一个终端服务，通过这个终端服务，可以使两人或多人使用网络即时的传递文字信息、档案、语音与视频交流，通信的形式分为电话即时通信、手机即时通信和网站即时通信。手机即时通信代表是短信，如飞信等，网站、视频即时通信的代表如 QQ、MSN、Gtalk 等。

不管哪一种解释，即时通信说明了一个核心的主题，那就是即时通信是用于实时的信息交换，这些信息包括互联网上一切可以用于传输的数据形式，语音、视频、图像、文本、数据流等。

注意：由于 NetLingo 在互联网专业词汇释义方面具有比较大的影响，因此基本上都以 NetLingo 对即时通信的定义为基准。同时 NetLingo 是在线更新的词典，它会经常针对互联网技术的变化对词汇释义进行修改，在本文中对即时通信的定义就是最新的，如果有更新的解释出现，以新的定义为标准。

8.1.2　即时通信的发展历程

即时通信（IM）的出现和互联网有着密不可分的关系，从技术上来说，IM 系统完全是基于 TCP/IP 网络协议族实现的，而 TCP/IP 协议族是整个互联网得以实现的技术基础。

最早期的即时通信雏形，可以追溯到芬兰人 Jarkko Oikarinen 于 1988 年发明的一种网络聊天协议 IRC（Internet Relay Chat），该协议仅支持文本聊天，并且也不支持好友列表

的概念。在 1996 年的时候，以色列两个工程师开发年第一个 IM 产品——ICQ，ICQ 喻意为 I seek you 的意思，ICQ 发明后，即时通信的技术和功能开始基本成型。如图 8.1 所示的就是 ICQ 的系统界面。

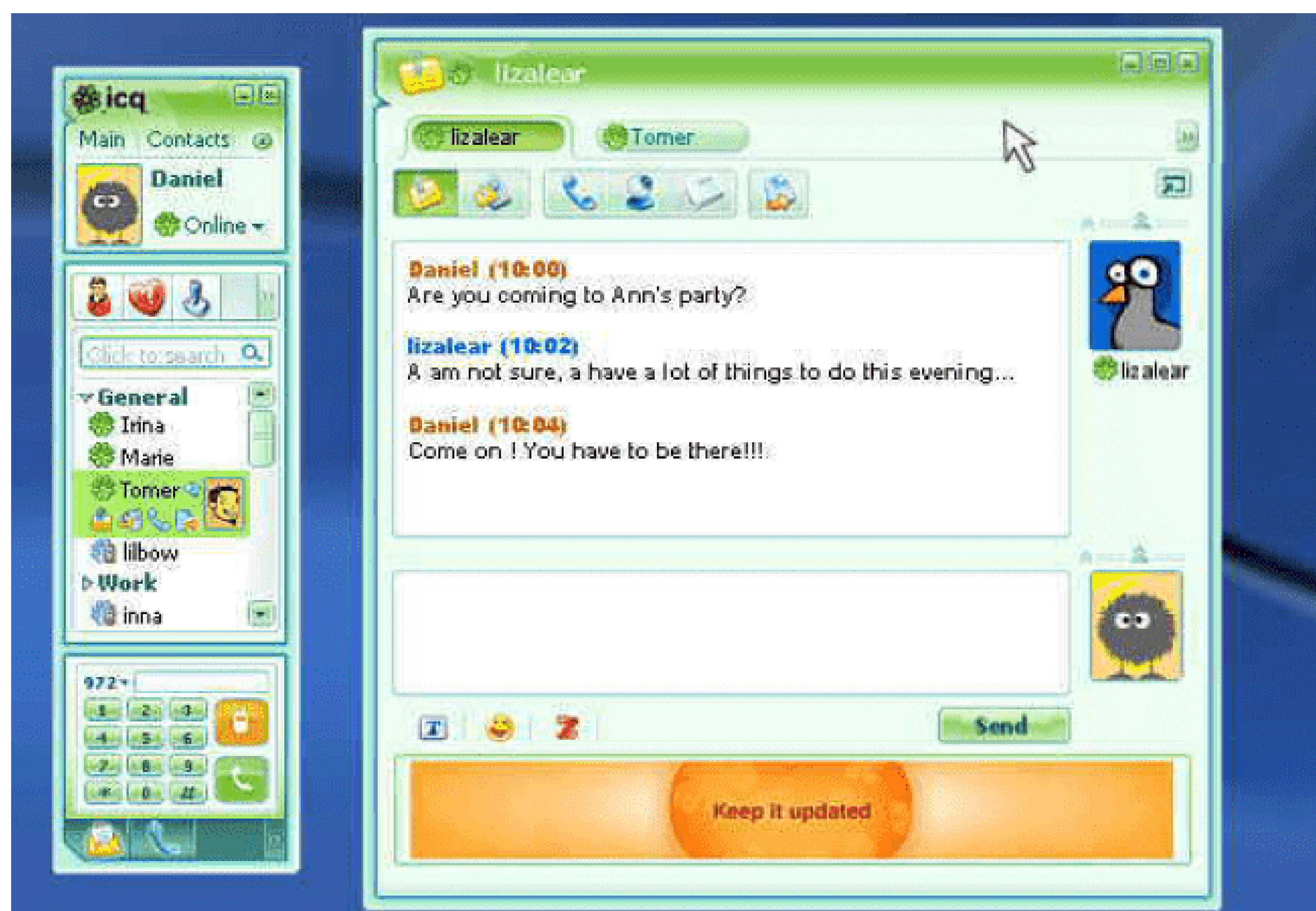


图 8.1　ICQ 系统的运行界面图

ICQ 的作为即时通信的始祖，它的开发成功随即引领着即时通信技术进入了一个高速发展的时期，在这一时期，各类 IM 软件百花齐放、百家争鸣，下面就简要的说一下一些主要的 IM 软件的发展历程。

1．ICQ

ICQ 作为世界 IM 的鼻祖，在上文已经提到过了，需要说明一点的是，ICQ 在开发之初其操作十分复杂，功能也比较单一。当时的 ICQ 由于没有找到什么盈利模式，后来就卖给了美国在线 AOL（美国新闻门户）。不久后 AOL 高层也意识到 ICQ 的复杂，开发出了 ICQ Lite 版本，这一版本的界面非常简单易用，有点像现在的 Gtalk。ICQ lite 上也开始有了 Google 搜索引擎链接，可以说 Google 能有今天的风光，ICQ 也算是帮了大忙吧。

2．雅虎通

“雅虎通”也是出现较早的、在全球影响较大的 IM 系统，因为它的“东家”是鼎鼎大名的雅虎，所以在雅虎的背后运作下，雅虎通发展得很好。据艾瑞 2006 年调查，雅虎通在美国 IM 市场大约占 18%左右，比 ICQ 少一些。雅虎通的系统界面如图 8.2 所示。

雅虎通很突出的一点还在于，它第一次成功将 IM 系统平台嵌入到商业网站上运行，著名的案例就是雅虎和新浪合资的一拍网。在一拍网上的用户可以用嵌入网页的雅虎通自由通信。现在我们使用淘宝和 eBay 时，其网页上也嵌入了相应的 IM，如阿里旺旺等，它们应用得都非常成功，极大方便了在线用户之间的交流。

3．MSN

牛气冲天的微软的 MSN，不说家喻户晓，但至少常接触电脑的 IT 人士、职场人员是

再熟悉不过的了。就技术层面而言，它是一个很普通的 IM 软件，跟其他的 IM 软件相比，几乎没有什么值得特别称道的地方。一个通用的即时通信系统的功能 MSN 都具备，用起来也很顺手、很方便。微软 MSN 不同的版及界面截图如图 8.3 所示。



图 8.2　雅虎通的系统运行界面



图 8.3　MSN 不同的版本及界面运行截图

在今天的互联网上，MSN 之所以能这么流行，主要跟它的销售行为和产品战术有关。MSN 与 Windows 系统捆绑起来，无缝嵌入到 Windows 的其他服务中，一站式的服务，完全兼容的技术，极大方便了用户的使用。正是这个策略让微软战胜了无数桌面软件的元老，包括 Netscape。

在美国，MSN Massager 的市场份额比雅虎通少一点。在中国，由于和 Windows 的捆

绑，MSN 有很多的高端用户。MSN 的定位也是商务用户，比如，MSN 需要用邮箱登录、MSN 不能像 QQ 一样在一台电脑登录多个……这些特点非常符合商务用户的使用习惯，以至于有不少人把使用 MSN 当成一种“时尚”。

🔔注意：MSN 这个定位商端用户的策略，曾让腾讯焦急万分，因为很多原来是学生的 QQ 用户随着年龄增长、工作，越来越多的人用 MSN 谈商务，这样 QQ 就是为 MSN 输送和培养高级用户了。所以后来，腾讯不得不做了个不伦不类的 TM，TM 在很长一段时间被公认是失败的产品，但就现在来看，TM 还是很不错的。

还需要提一点的就是，最新版 MSN live 开放了 SDK 开发包，中国有专门的公司利用这个开发包开发类似于 MSN“机器人”程序。MSN 用户可以将这些机器人加为好友，进行聊天、看电影、玩小游戏、天气预报等，相当于是 MSN 的增值业务。这也是一种新的盈利模式。

🔔注意：ICQ、雅虎通、MSN 现在已经实现互联互通，也就是说使用这 3 款软件相互之间可以通信。

4．Skype

IM 的发展说到这里，一个很重要的“人物”出现了，他就是 Skype。Skype 最初是由两个 P2P 技术的元老级人物开发的，不同于以上任何一种 IM。它的技术、特性、功能价值，很难简单地概括，先在这里留点悬念，后文会有详细的关于 Skype 技术的讲解。

5．Gtalk

Gtalk（Google talk）是 Google 于 2005 年底开发推出的自己的一款 IM 产品，Gtalk 刚刚出来的时候在它的网页上有这样一句广告语：翻译成中文意思大概是“有些 IM 很 cheap，但是 Gtalk 很 Free！”。

🔔注意：至于这个广告语是什么意思，读者可以自己去想象，但笔者感觉，这与 Skype 有关。

当前 Gtalk 版本有两种形式，一种是下载安装的桌面版 Googel Talk，另一种是网络版的 Gtalk。桌面版 Googel Talk 主要的功能有，即时通信、计算机与计算机间（PC-to-PC）的免费语音呼叫、发送和接收语音邮件、在桌面上接收 Gmail 通知等。至于 Gtalk 的详细功能和使用方法，有兴趣的读者可以到 Google 的官方网站下载试用，研究一下就知道了。

6．QQ

QQ 是我们最熟悉的即时通信平台，从其诞生至今发展非常迅速。自 1999 年 2 月第一款 QQ 软件推出以来，注册用户数飞速增长。2006 年腾讯公司公布的第四季度即时通信注册用户数达到 5.8 亿。2009 年 2 月，腾讯公司 QQ 同时在线用户数突破 5000 万人。同年 4 月，腾讯公司的字母 QQ 商标被国家工商行政管理总局认定为驰名商标。

这所有数据足以说明 QQ 是当今即时通信领域无可争议的明星。QQ 最开始称作 OICQ，

大概是“Oh！I seek you”的意思，其技术来源于 ICQ。值得一提的是，QQ 是世界第一个找到 IM 商业模式的运营商，它以 QQ 即时通信为平台，现已发展形成了即时通信业务、网络媒体、无线互联网增值业务、互动娱乐业务、互联网增值业务、电子商务和广告业务 7 大业务体系，并初步形成了“一站式”在线生活的战略布局。QQ 在中国市场上的巨大成功，足以说明即时通信系统的内在价值和潜力。

其他的还有很多即时通信系统，如网易泡泡，它与 QQ 竞争的杀手锏就是有偿免费手机短信，通过用户在线时长换取“泡币”，一定数量的“泡币”可以换取免费短信。这个办法在一定程度上还是吸引了很多年轻人，泡泡也是 2004 年以前排名第 3 的 IM。还有 UC，它是国内一家公司开发的视频和语音都非常好的 IM，后来被新浪收购后就没有下文了。另一个要说的就是淘宝“阿里旺旺”，它是嵌入到“淘宝网站”网页中的 IM 系统，旺旺的表情营销可以说是互联网营销的经典案例，效果确实出奇的好。“淘宝旺旺”与“一拍”的雅虎通，这种在网页中嵌入 IM 的思想，可以说是对 C2C 市场最大的贡献之一。

注意： C2C 即 Consumer To Consumer 的意思，是一种消费者对消费者的电子商务模式。C2C 的商务平台就是通过为买卖双方提供一个在线交易平台，使卖方可以主动提供商品上网拍卖，而买方可以自行选择商品进行竞价。

7. Linux下常用的即时通信系统

以上提到的几款即时通信系统，常部署在 Windows 环境下，还有一些 Linux 系统下常用到的即时通信系统，下面简要地提一下。

- amsn：amsn 对应的是 Windows 下的 MSN Messenger。
- Licq：Licq 对应的是 Windows 下的 ICQ 客户端，软件可以通过新立德安装。

注意： 新立德的软件安装方式是 Linux 的 Ubuntu 系统下的一种软件安装方式，请参考 Ubuntu 系统的相关知识。

- EVA：对应的是 Windows 下的 QQ 软件，在没有 qq linux 及 qq linux 1.0 不完善时，它是大部分 Linux 用户的选择。
- XChat：XChat 对应的是 Windows 下的 IRC 公众聊天讨论组，类似于聊天室的功能。
- Gaim/Pidgin：异常强大的 Linux 多通信协议软件，支持 AOL instant messenger、gadu-gadu、ICQ、IRC、jabber、MSN、novell group wise 、penNAP、yahoo !messenger、zephyr、SILC、google talk 还有 QQ 等，几乎无所不能是 Linux 中常用的一个即时通信平台。如图 8.4 所示的就是 Pidgin 所支持的即时通信软件列表。
- qq for linux 1.0：腾讯公司正式推出了 Linux 版本的 IM。qq for linux 1.0 版本目前还有很多不完善的地方需要改进。

至于其他的即时通信软件，如 mymeet、imo（互联网办公室）、飞信、paltalk，LumaQQ、mICQ 等，也都是常用的即时通信平台，读者可自行查阅相关资料以了解它们的特点、功能等。

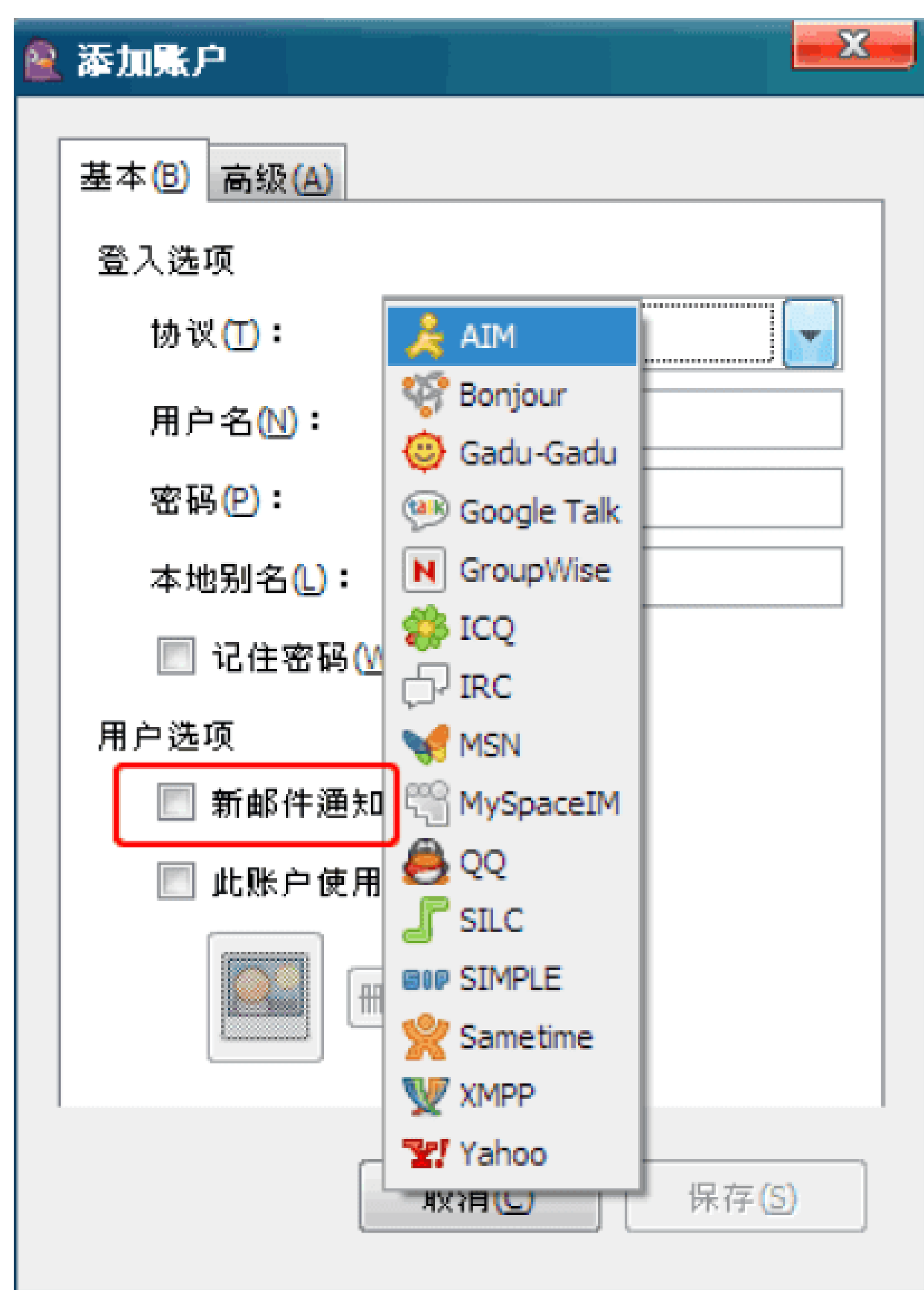


图 8.4　Linux 系统下 Pidgin 所支持的即时通信软件列表

注意：以上讲了那么多即时通信软件，但是它们之间却没有一个统一的标准。也就是说，能够与你进行即时通信的人必须使用和你一样的 IM 系统才行。简单地说就是，一个使用 QQ 的人和一个用 MSN 的人是无法相互通信的，尽管他们用的都是 IM 系统。

8.1.3　即时通信系统是怎样工作的

即时通信系统具有一个完整的工作流程，要让一个即时通信系统工作起来，最基本的需要做以下几步的工作。

- 确定自己的系统平台、硬件型号等，根据这些信息来选择一个能够满足自己需要的即时通信软件。
- 下载即时通信软件的安装包。现在几乎所有的即时通信软件都是免费的，有的还开源，可以直接到它的官方网站进行下载。
- 安装下载的即时通信软件。安装成功后，需要从此软件的服务提供商那里注册个人信息并获得唯一的名称。
- 根据注册的名称，登录中心服务器了，这时你的 IM 系统才会显示你处于可用状态。
- IM 系统成功启动后，根据此即时通信系统提供的功能，进行各类操作。

不同的 IM 系统根据应用的侧重点不同，其功能也不尽相同（下文会讲到它们的功能），从一个用户的角度出发，使用一个即时通信系统时，基本的操作可由如图 8.5 所示的用例图来表示。

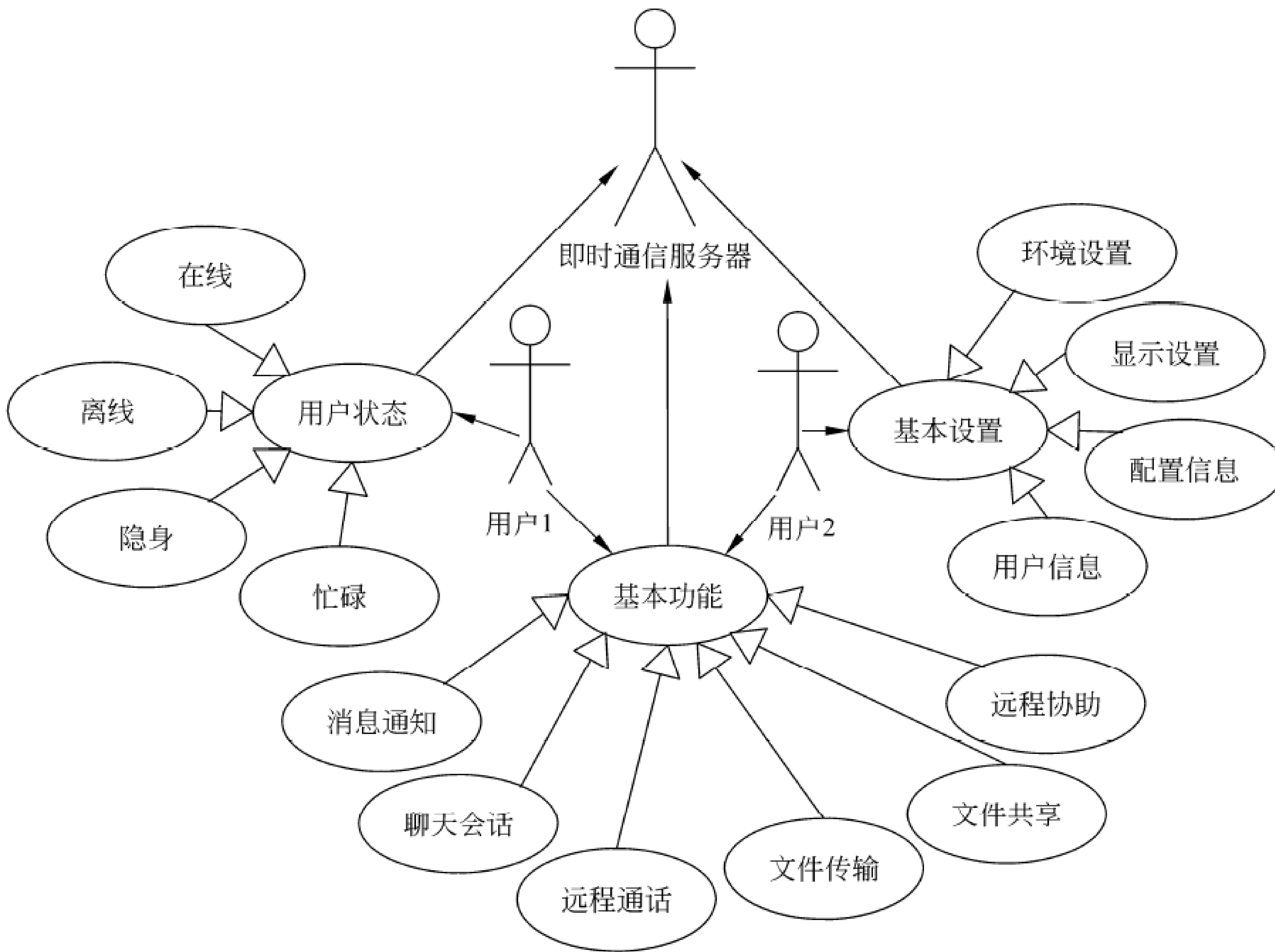


图 8.5　用户使用即时通信系统的基本用例图

用例图 8.5 展示的是一个普通用户使用即时通信系统时一些最基本的操作，如更改状态、消息交互、资源传输等。

8.1.4　即时通信系统的基本原理和工作流程

最初的即时通信系统 ICQ 是以网络聊天协议 IRC（Internet Relay Chat）为基础开发而成的，该协议虽然支持文本聊天，但功能上还是比较单一。所以，后来不同的厂商开发自己的 IM 系统的时候所采用的协议都有较大的差异，但我们仍然可以从一个提供最基本服务的 IM 系统开始来描述 IM 的技术原理。不管目前产品的新功能如何丰富，它必须遵循这些基本原理和结构。

注意：当前很多即时通信软件所采用的协议都是私有甚至加密的。到目前为止世界主要的 IM 服务运营商 AOL（American Online：美国在线）仍然没有公布其主要即时通信产品 AIM（American Instant Messenger）的专用协议。

1. IM的通信原理

即时通信系统的通信过程是建立在 TCP/IP 和 UDP 协议的基础的，它们都是低层的 IP 协议上的两种通信传输协议。在即时通信系统中，TCP/IP 协议，常以数据流的形式，将传输数据经分割、打包后，再通过两台机器之间建立起的虚电路，以连续的、双向的、严格保证数据正确性的要求进行文件传输。而 UDP，则以数据报的形式，对拆分后数据的先后

到达顺序不做要求，进行不可靠的文件传输。在即时通信系统中，具体的通信过程如下。

- ❑ 在即时通信系统中，一般都使用 UDP 协议进行发送和接收“消息”的。
- ❑ 当你的机器安装了 IM 以后，实际上，你既是服务端（Server），又是客户端（Client）。
- ❑ 当你登录 IM 时，你的 IM 系统作为 Client 连接到服务公司的主服务器上。
- ❑ 当你“看谁在线”时，你的 IM 系统又一次作为 Client 从 IM 的 Server 上读取在线网友名单。
- ❑ 当你和你的在线伙伴进行聊天时，如果你和对方的连接比较稳定，你和他的聊天内容都是以 UDP 的形式，在计算机之间传送。
- ❑ 如果你和对方的连接不是很稳定，IM 的服务器将为你们的聊天内容进行“中转”。

总地来说，以上使用即时通信系统的过程可以用如下 3 句话来综合的描述。

- ❑ 用户首先从 QQ 服务器上获取好友列表，以建立点对点的联系。
- ❑ 用户（Client1）和好友（Client2）之间采用 UDP 方式发送信息。
- ❑ 如果无法直接点对点联系，则用服务器中转的方式完成。

除了一些特别的即时通信软件（如，Skype）外，其他即时通信软件的通信原理与此大同小异，几乎都遵循以上这几个过程。

注意：TCP/IP 是（Transmission Control Protocol/Internet Protocol）的简写，中文译名为传输控制协议/网际协议，又叫网络通信协议，这个协议是 Internet 最基本的协议，是 Internet 国际互联网络的基础。而 UDP，用户数据报协议（User Datagram Protocol），是 OSI 参考模型中一种无连接的传输层协议，提供面向事务的简单不可靠信息传送服务，是一个简单的面向数据报的传输层协议。UDP 协议基本上是 IP 协议与上层协议的接口，适用端口分别运行在同一台设备上的多个应用程序。

2．即时通信系统的执行流程

从即时通信软件的角度来看，要执行一个完整的通信流程需要以下几个基本而必须的步骤。

（1）用户 A 输入自己的用户名和密码登录即时通信服务器，服务器通过读取用户数据库来验证用户身份。如果用户名、密码都正确，就登记用户 A 的 IP 地址、IM 客户端软件的版本号及使用的 TCP/UDP 端口号，然后返回用户 A 登录成功的标志，此时用户 A 在 IM 系统中的状态为在线（Online Presence）。以 QQ 为例，用户的登录过程如图 8.6 所示。

（2）根据用户 A 存储在 IM 服务器上的好友列表（Buddy List），服务器将用户 A 在线的相关信息发送到也同时在线的即时通信好友的 PC 上，这些信息包括在线状态、IP 地址、 IM 客户端使用的 TCP 端口（Port）号等。即时通信好友 PC 上的即时通信软件收到此信息后将在 PC 桌面上弹出一个小窗口予以提示。

（3）即时通信服务器把用户 A 存储在服务器上的好友列表及相关信息回送到他的 PC 上，这些信息也包括在线状态、IP 地址、IM 客户端使用的 TCP 端口（Port）号等信息，用户 A 的 PC 上的 IM 客户端收到后，将显示这些好友列表及其在线状态。如图 8.7 所示为 QQ 好友列表状态信息。



图 8.6　QQ 用户的登录过程



图 8.7　QQ 好友的列表状态

（4）如果用户 A 想与他的在线好友用户 B 聊天，他将直接通过服务器发送过来的用户 B 的 IP 地址、TCP 端口号等信息，直接向用户 B 的 PC 发出聊天信息。用户 B 的 IM 客户端软件收到后显示在屏幕上，然后用户 B 再直接回复到用户 A 的 PC 上，这样双方的即时文字消息就不通过 IM 服务器中转，而是通过网络进行点对点的直接通信，称为对等通信方式（Peer To Peer）。

在商用即时通信系统中，如果用户 A 与用户 B 的点对点通信由于防火墙、网络速度等原因难以建立或者速度很慢。IM 服务器还提供消息中转服务，即用户 A 和用户 B 的即时消息全部先发送到 IM 服务器，再由服务器转发给对方。

早期的 IM 系统，在 IM 客户端和 IM 服务器之间通信采用 UDP 协议，UDP 协议是不可靠的传输协议，而在 IM 客户端之间的直接通信中，采用具备可靠传输能力的 TCP 协议。随着用户需求和技术环境的发展，目前主流的即时通信系统倾向于在即时通信客户端之间、即时通信客户端和即时通信服务器之间都采用 TCP 协议。

8.1.5　即时通信的基本功能及应用

即时通信软件除了可以实时交谈和互传信息，不少还集成了数据交换、语音聊天、网络会议、电子邮件的功能。那么作为一个即时通信系统，它可以完成的哪些最基本而又常用的功能呢？

1．文字聊天

聊天功能是 IM 软件最基本、也是最重要的功能，基本上每一种 IM 软件在这个功能上

的操作都差不多：如果用户想与联系人进行聊天，可以双击 IM 中联系人的头像，在弹出的对话框中输入文字信息发送即可。QQ 的特点是可以给不在线的朋友发送信息，对方下次上线的时候可以收到，MSN 虽然不具备这样的功能，但是它在聊天过程中可以使用各种漂亮的表情图标为聊天添加了不少情趣。

2．语音聊天

如果打字聊天的方式已不能满足，QQ 还提供了“二人世界”里的实时语音聊天。首先需要有音箱或者耳机、麦克风，然后就可以向你的网友发送连接到二人世界的请求。通过后双方不仅可以用文字聊天，还可以直接讲话。此外 QQ 还有传送语音功能，利用此功能可以传送语音信息。首先单击在线好友的头像，选择“传递语音”选项，然后就会弹出一个对话框，录音以后就可以发送了。

3．传送文件

IM 软件能点对点地传输文件，有时候利用此功能要比使用 E-mail 还方便许多，当然此项功能必须在对方在线时才能使用。在 QQ 的好友头像上右击，在弹出的快捷菜单中选择“传送文件”，选定要传送的文件，单击“发送”按钮，等待对方接受请求。此外，ICQ 的文件传送功能还支持类似断点续传的功能，不必担心文件传送过程中发生突然中断的情况。

4．拨打电话

在 MSN Messenger 中提供了 PC-PHONE 的拨打电话功能，可以在 MSN Messenger 软件主窗口中，单击操作窗口“我想”下面的“拨打电话”或者右击要呼叫的人的名字，在弹出的快捷菜单中选择“拨打电话”选项，如图 8.8 所示。

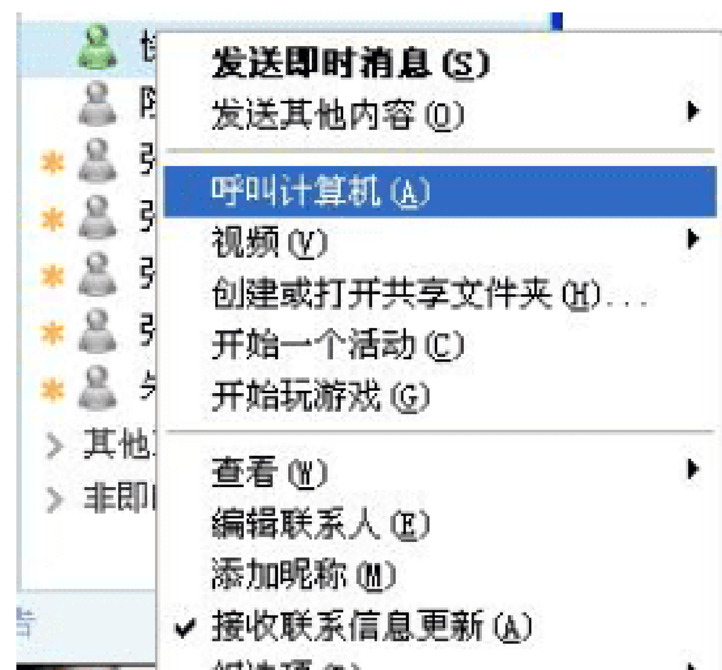


图 8.8　MSN 中的语音呼叫功能

MSN 中可以开启拨叫电话功能。但在进行电话呼叫之前必须注册语音服务提供商。由于 MSN Messenger 在国内暂时还没有开通这项业务，所以国内的用户暂时无法使用。

5．远程协助

远程协助是在 Windows XP 中引进的新概念，是 Windows Messenger 独有的功能。远程协助可以将电脑的控制权分享给对方以便于对寻求协助者提供帮助，通过它，对方可以很容易地控制寻求协助者的桌面。如图 8.9 所示为 QQ 中的远程协助功能。

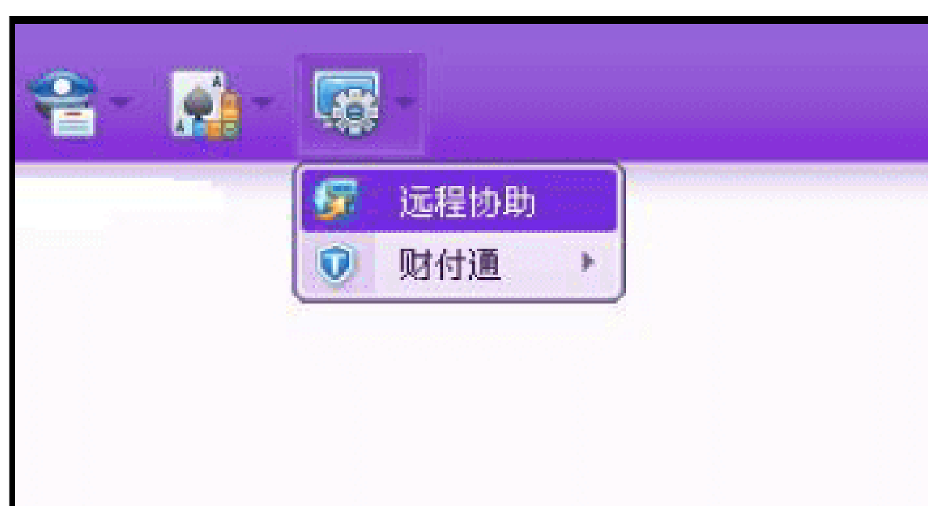


图 8.9　QQ 中的远程协助功能

远程协助的功能主要体现在应用程序共享、远程协助、白板共享、寻求远程协助等方面。由于这一功能非常强大，在寻求协助的过程中系统会多次提醒并给出选择，用户在使用这一功能时需多加小心，确认对方是否可靠。

6．视频聊天

如果你的网速够快，又有摄像头，完全可以用 IM 软件来代替 Net meeting 了，在聊天的同时，不仅可以通话，还可以看到对方的图像，表情，倍感亲切，给您带来一份全新的感受。Windows Messenger 对视频聊天功能支持很好，设置和使用也非常简单，右击好友中“开始视频对话”选项就可完成操作，非常方便。

7．邮件辅助

IM 和 E-mail 是在网上最常用的两种工具，如今不少 IM 软件将两者做了完美的结合。在 QQ 中你可以直接给自己的好友发邮件，而无须再输入 E-mail 地址；此外对于自己的信箱，QQ 还有检查新邮件功能。在“系统参数”中设置自己的 E-mail，填好 POP3 地址，可以选择定时检查时间，QQ 就会自动检查是否有新邮件到达。对于 MSN Messenger 来说，它的邮件功能就更强大了，使用 MSN Messenger 必须有一个邮件账号，每次当你的 MSN Messenger 登录成功时，在右下角自动弹出一个窗口，里面写有该 E-mail 账户内的信件状况。在使用过程中如果您的邮箱中有了新邮件，马上会冒出一个提示窗。

8．发送短信

目前 IM 与各种移动终端设备的结合也越来越多。使用 QQ 向手机发送短信需要手机开通移动 QQ 服务。单击对方头像图标，在打开的快捷菜单中选择“手机短信”命令，在打开的对话框中输入信息，然后单击发送即可完成，这时对方的手机就可以收到一个消息。对方的手机收到你发来的短信后还可以回复，这时你的 QQ 会弹出“查看手机短信”的窗口，非常方便。现在也有专门的用于和手机进行通信的 IM 系统，如飞信等。

9．资讯发布

大部分的即时通信系统都有新闻展示这一功能，会随 IM 的启动而弹出相应的新闻展示页面，内容丰富、种类详细，如新闻、IT 科技、证券、体育、娱乐……各类不同的资讯。如图 8.10 所示，就是 QQ 系统发布资讯的界面。

想看哪方面的新闻，就单击相应的图标即可，IM 系统会自动提取出当日新闻标题。通过这些标题，可以快速地选择出自己感兴趣的新闻，单击它就可以调用浏览器读取了。这样通过 IM 系统可以很方便地阅读有关内容，节省了查找时间、提高了浏览效率。



图 8.10　QQ 中的资讯发布功能

各种即时通信软件根据应用环境和针对用户群的不同，其功能和特色也不一样，但是作为一个即时通信系统而言，以上的功能都是基本而且必备的，要开发一个 IM 系统，这些功能也是必须要考虑进去的。

10．即时通信的商业应用

从商业盈利的角度讲，即时通信系统可以带来巨大的产业价值，最经典的商业应用应当属于腾讯 QQ 了。就 QQ 本身而言，它完全就是一个纯粹的即时通信软件，但是在市场的运作下，现在已完全成为一个巨大的产业，并由此带动的产业链，其带来的价值不可估量。

从商业应用的角度来讲，即时通信系统已经成为是企业管理、企业交流不可缺少的重要平台，很多企业都开发了自己内部的即时通信系统，用于即时的发布消息、交流资讯和共享资源。

除此以外，即时通信系统在商业应用上的成功案例举不胜举，总地来说，即时通信在未来的商业发展具有极大的潜力和商业价值。

8.1.6　即时通信系统发展的机遇与存在的问题

随着互联网的普及和宽带技术的发展，以 P2P 技术为核心的软件产品正在为越来越多的网民所接受和喜爱。今天，P2P 技术的发展已经不可阻挡，其应用范围也越来越广泛，以 Skype 为代表的即时通信系统有着广阔的发展前景和巨大的发展潜力。

1．从MSN的中断看即时通信在中国的发展前景

由于北亚地区海底光缆发生故障，即时通信软件 MSN 于 2009 年 8 月 16 日出现大面积掉线，17 日下午再次出现登录故障。许多网友反映无法登录，或者登录后持续掉线。

MSN 再次发生故障使众多用户手足无措，这是在很多人意料之中的事情，因为很多 MSN 的高端用户都发出过这种忧虑。在即时通信系统快速发展的今天，MSN 在中国作为白领象征的统治时代何时才能结束呢？中国人自己的 MSN 何时才能出现？

这一次 MSN 大规模掉线像是一次预演——更大规模、更深层次的破坏，是因这次掉

线而造成的即时通信危机。所幸这是一次突发性的、偶然的事故，并非有预谋的、有目的的破坏。如果真的是一次非突发事件的话，那后果将不堪设想。

这种危险，不仅警告我们：中国该开发“中国 MSN”了！也昭示着即时通信在中国的发展空间很大，前景很广。

也许有人会说，中国的即时通信工具够多了，还用开发吗？QQ 的确强大，但它的强大仅限于用户群的庞大和功能的繁多。正是这种繁杂的功能和夹杂着的太多商业元素，使得 QQ 难以适应高端的商务需求，至于其他的即时通信工具，它们比 QQ 又能强多少呢？

所以，开发一款有别于现行即时通信工具的“中国 MSN”已经迫在眉睫，否则未来的网络世界发生突然事故的情况下，网络通信将无从谈起。如果我们依然痴迷于 MSN，那未来谁要是切断一根电缆的话，我们很多人就寸步难行了。至于漏洞百出的国内现有即时通信工具，我相信它们更会让我们与安全说拜拜。

单就这一点来说，在中国真正能满足高端用户需要的 IM 几乎没有。腾讯的 TM 虽然有这份心，但就目前的反应来看，实在难当重任，至于其他的即时通信平台，不提也罢。与国外相比，我们已经有不少地方落后了，如果再在即时通信方面受制于人的话，未来会很危险的。所以，中国该开发“中国 MSN”了！读者可以想象一下，如果有一款中国人自己的、安全而又高效的 MSN，又会带来怎样潜在的机会呢？

2．即时通信还存在问题

就目前来看，即时通信存在两个大的问题。

一是协同工作问题，AOL 的用户不能与 MSN 的交流，MSN 的用户不能同 QQ 的交流。如果你要同某人即时通信，则大家必须是使用相同的服务。如果你的客户有 10 个人，每个人都使用不同的即时通信软件，那么你的机器上至少得装上 10 款软件，这是多么可怕的事情。不过，从将来的发展来看，即时通信系统如果不能做到一家独大的话，相互之间的整合与协同必是大势所趋。

即时通信的另一个问题，就是安全问题，这也是个最头疼、最难克服的问题、上文中已经分析 IM 系统基本的通信方式，用户之间消息的传递是以 UDP 的方式进行的，UDP 本身就是不可靠的传输协议，随时都有可能发生数据的丢失。另外一点是，在数据发送的时候，有时候一些数据是需要服务器进行转发的，而很多的 IM 系统并没有对这些需要转发的数据进行任何加密。也就是说，如果你要向你的密友说一句悄悄话的时候，很可能，你的“悄悄话”要先到服务器那里逛一圈，再通过服务器送到你的“好友”那里。如果我就在服务器旁边的话，你的所谓的“悄悄话”岂不是被我看得一清二楚？

这种安全问题是由当前即时通信系统本身的机制造成的，如果要克服这种问题，那么整个即时通信的原理和机制都要从根本上被颠覆。也就是说，消息传输不再用 UDP，信息的发送无须服务器中转，所有的通信过程都在客户端与客户端之间完成，就是以 Peer to Peer 的方式来实现两个用户之间的交流。而这恰恰就是 P2P 技术。

利用 P2P 技术的即时通信系统，不仅可以有效地解决安全问题，也最大限度地减轻了集中式服务器的负担、节约了成本同时也增加了系统的可扩展性和健壮性。从下节开始将会重点讲解 P2P 技术在即时通信方面的应用。

8.2　基于 P2P 技术的即时通信系统——Skype

尽管即时通信软件的广泛流行，但是多数此类软件仍然采用的是传统的 C/S 模式，极大的在线用户数量需要服务器进行大量的用户管理和通信协调工作，从而制约了此类系统的发展。而 P2P 技术正是对传统 C/S 模式的一种颠覆。P2P 作为一种网络模型，其中所有的结点（客户端）是对等的，各结点具有相同的责任和能力，能够协同完成任务。

将 P2P 技术与即时通信系统结合起来，能较大地改善现有即时通信系统的维护性能高，可扩展性差的缺点。P2P 技术在即时通信领域最经典、最成功的应用就是 Skype。

8.2.1　Skype 简介

你今天 Skype 了吗？当 Skype 从一种技术变成实际应用的时候，一夜之间它就成了许多人每日生活中不可或缺的沟通工具。Skype 为联网带来了一个全新的通信时代，Skype，是支持语音通信的即时通信软件，由 KaZaA 开发人员所研发，采用 P2P 的技术与其他用户连接，可以进行高清晰语音聊天，联机双方网络顺畅时，音质可能超过普通电话。如图 8.11 所示的就是 Skype 软件的标志。



图 8.11　Skype 软件的标志

Skype 和 ICQ、MSN 很像，是一套即时通信软件，不同的是，Skype 采用点对点交换方式进行信息的传输，且特别增强语音传输功能，让网友不但可以透过 IM 平台交换文字信息，还可以用 CD 唱片的音质水准进行语音沟通，如同打电话一般。

Skype 是一种软件，可以在网络上免费下载。下载后的 Skype 软件，可以快速简便地装入到电脑。只需简单的信息注册，在数分钟之内，便可以通过 Skype 与你的朋友通电话。Skype 通话具有非常好的音质，双方通话采用密码传送方式，高度安全可靠。最好的一点是，Skype 无须重新配置防火墙或路由器便可正常工作！

Skype 最重要的特点就是它具有 VoIP 的功能，利用 P2P 技术在互联网上进行语音传输，使用 Skype，可以使人们能够在数分钟之内在世界上的任何角落拨打免费高质量电话。除了网内互打，也可以透过 Skype 拨打电话给只有固网电话或移动电话的朋友，这无疑是 VoIP 发展史上划时代的里程碑。“下载了一个 Skype 软件之后，我就知道完了。Skype 通话的质量好极了而且是免费的，一切都结束了”。美国联邦通信委员会主席麦克尔·鲍威尔曾

感叹道。

注意：关于 VoIP 的相关知识，请参阅下文的讲解。

正如麦克尔·鲍威尔所说，Skype 的确优秀，可以说，它是迄今为止公认的最成功的 P2P VoIP 软件，在可用性和通话质量方面都非常出色。它可以无缝地穿越防火墙和 NAT（网络地址转换）设备，用户无须进行任何配置。在语音质量方面，在拨号连接的带宽下就可获得传统电话的语音质量，这也比 MSN 或 Yahoo messenger 等即时消息软件要出色。它采用私有协议，所有的语音数据都进行端对端的加密，所有用户数据都是分布方式存储，同时它也支持即时消息和会议功能。

总地来说，Skype 出现的重大意义源于它的颠覆性本质，它证明了使用现有的 Internet 带宽资源进行高质量语音服务的可行性。同时，它以铁的事实证明，语音通信可以是很便宜的。它的出现，使得传统电信行业不得不开始认真对待新一代通信机制的挑战。

8.2.2　Skype 的 VoIP 功能

VoIP 是 Voice over Internet Protocol 的缩写，指的是将模拟的声音信号经过压缩与封包之后，以数据封包的形式在 IP 网络的环境进行语音讯号的传输。通俗来说也就是互联网电话、网络电话或者简称 IP 电话的意思。VoIP 技术是目前互联网应用领域的一个热门话题，成为 2004 年全球互联网与电子商务十大趋势之一。

1．VoIP的简史

1995 年以色列 VocalTec 公司所推出的 Internet Phone，不但是 VoIP 网络电话的开端，也揭开了电信 IP 化的序幕。人们从此不但可以享受到更便宜、甚至完全免费的通话及多媒体增值服务，电信业的服务内容及面貌也为之剧变。

一开始的网络电话是以软件的形式呈现，同时仅限于 PC to PC 间的通话，换句话说，人们只要分别在两端不同的 PC 上，安装网络电话软件，即可经由 IP 网络进行对话。随着宽频普及与相关网络技术的引进，网络电话也由单纯 PC to PC 的通话形式，发展出 IP to PSTN（公共开关电话网络）、PSTN to IP、PSTN to PSTN 及 IP to IP 等各种形式，所有这些形式的共同点就是以 IP 网络为传输媒介，如此一来，电信业长久以 PSTN 电路交换网网络为传输媒介的惯例及独占性随着 VoIP 的出现也逐渐被打破。

注意：PSTN（Public Switched Telephone Network），公共电话交换网络的意思，是一种以模拟技术为基础的电路交换网络。我们日常生活中的电话网就是这种网络，相比而言，PSTN 一种比较旧式电话系统。

2．Skype的网络通信是怎样实现的

传统的电话网络，我们称之为模拟电话网或 PSTN 网，我们利用它来通话，其间所有的语音都是模拟信号在 PSTN 网上传输；而在 Internet 上，所有的数据，包括语音都是数字信号在 Internet 上传输。

现在 VoIP 电话可以穿透两大网络，使 Internet 上的数字语音信号和 PSTN 网上的模拟

语音信号相互转换。利用互联网来传送语音，无论何时何地，只要接入到 Internet 网络，就可以实现全球任何两点之间的语音通话。Skype 的 VoIP 功能，让你在享受快捷的呼叫连接、清晰的通话质量、免费的通话费用的同时，也保证了信息传递的安全、有效、正确。如图 8.12 所示的就是电话语音在 Internet 中传输的模型。

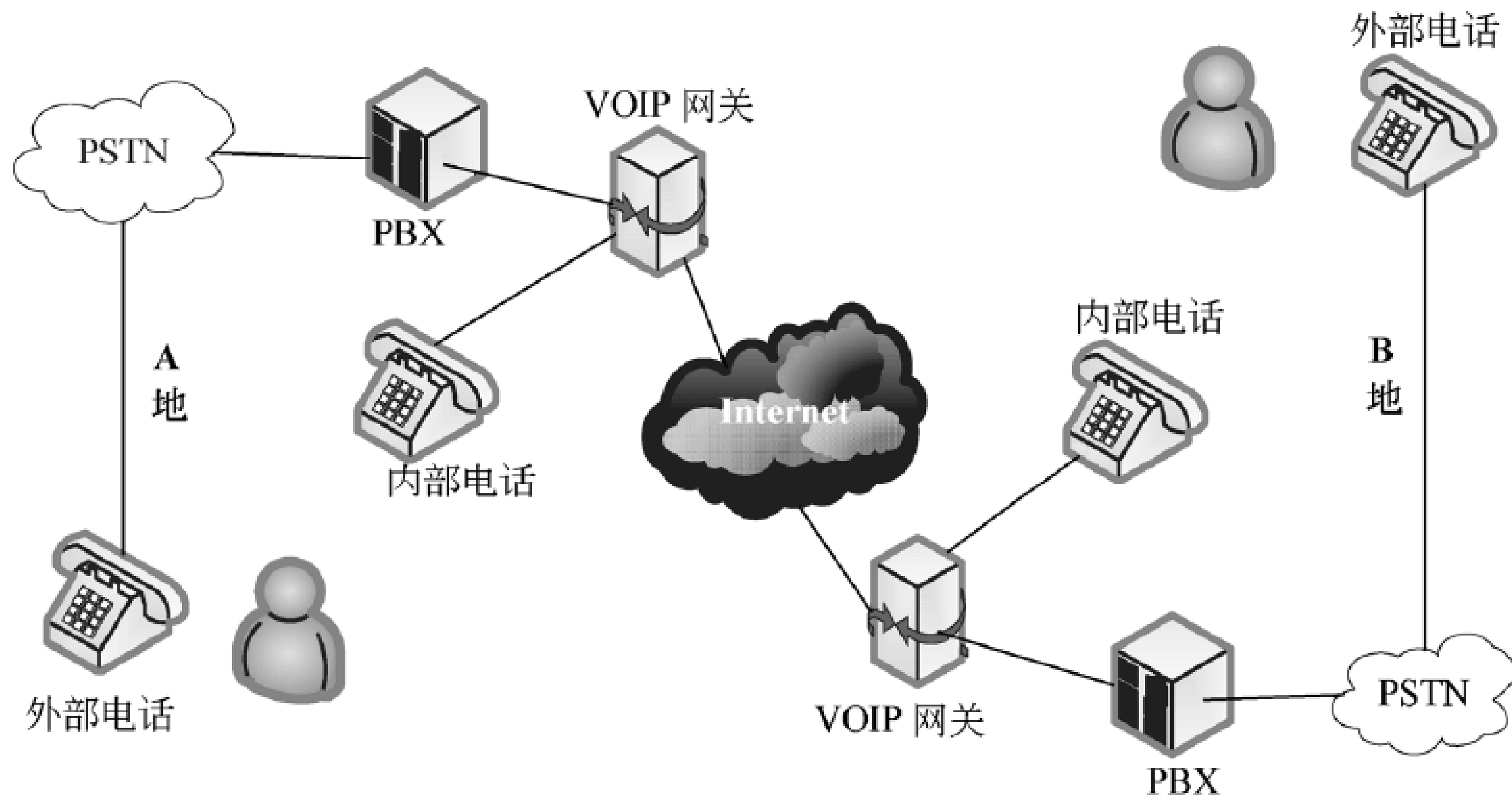


图 8.12　电话语音在 Internet 中传输的模型图

3．VoIP的工作原理

由 Voice over IP 的字面意义，可以直译为透过 IP 网络传输的语音信号或影像信号，所以 VoIP 就是一种可以在 IP 网络上互传模拟音讯或视讯的一种技术。简单地说，它是藉由一连串的转码、编码、压缩、打包等程序，好让该语音数据可以在 IP 网络上传输到目的端，然后再经由相反的程序，还原成原来的语音讯号以供接听者接收。

VoIP 的基本原理是：通过语音的压缩算法对语音数据编码进行压缩处理，然后把这些语音数据按 TCP/IP 标准进行打包，经过 IP 网络把数据包送至接收地，再把这些语音数据包串起来，经过解压处理后，恢复成原来的语音信号，从而达到由互联网传送语音的目的。VoIP 的工作原理如图 8.13 所示。

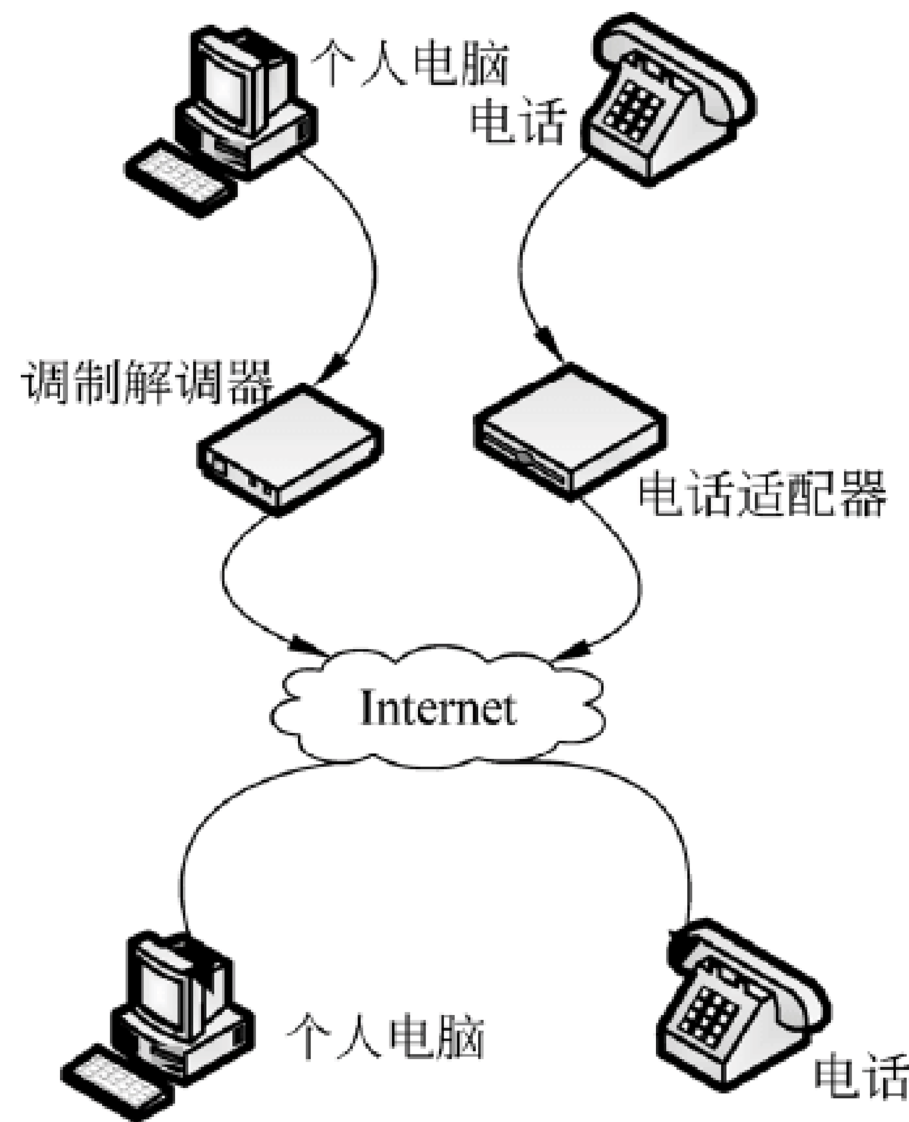


图 8.13　VoIP 的工作原理

IP 电话的核心与关键设备是 IP 网关，它把各地区电话区号映射为相应的地区网关 IP 地址。这些信息存放在一个数据库中，有关处理软件将完成呼叫处理、数字语音打包、路由管理等功能。

在用户拨打长途电话时，网关根据电话区号数据库资料，确定相应网关的 IP 地址，并将此 IP 地址加入 IP 数据包中，同时选择最佳路由，以减少传输时延，IP 数据包经 Internet 到达目的地的网关。在一些 Internet 尚未延伸到或暂时未设立网关的地区，可设置路由，由最近的网关通过长途电话网转接，实现通信业务。

进一步来说，VoIP 大致透过 5 道程序来互传语音信号。

（1）首先是将发话端的模拟语音信号进行编码的动作，目前主要是采用 ITU-T G.711 语音编码标准来转换。

注意： ITU-G.711 是一种由国际电信联盟（ITU-T）制定的音频编码方式，又称为 G.711 编码。

（2）第二道程序则是将语音封包加以压缩，同时添加地址及控制信息。

（3）第三道程序，也就是传输 IP 封包阶段，根据第二阶段得到的控制信息，将 IP 数据包传送到目的端。

（4）到了目的端，IP 封包会进行译码还原的作业，将 IP 数据还原成语音数据。

（5）最后，将语音数据转换成喇叭、听筒或耳机能播放的模拟语音信号。

在一个基本的 VoIP 架构之中，大致包含 4 个基本元素：分别为媒体网关器、媒体网关控制器、语音服务器和信号网关器，一个大型的 VoIP 架构图如图 8.14 所示。

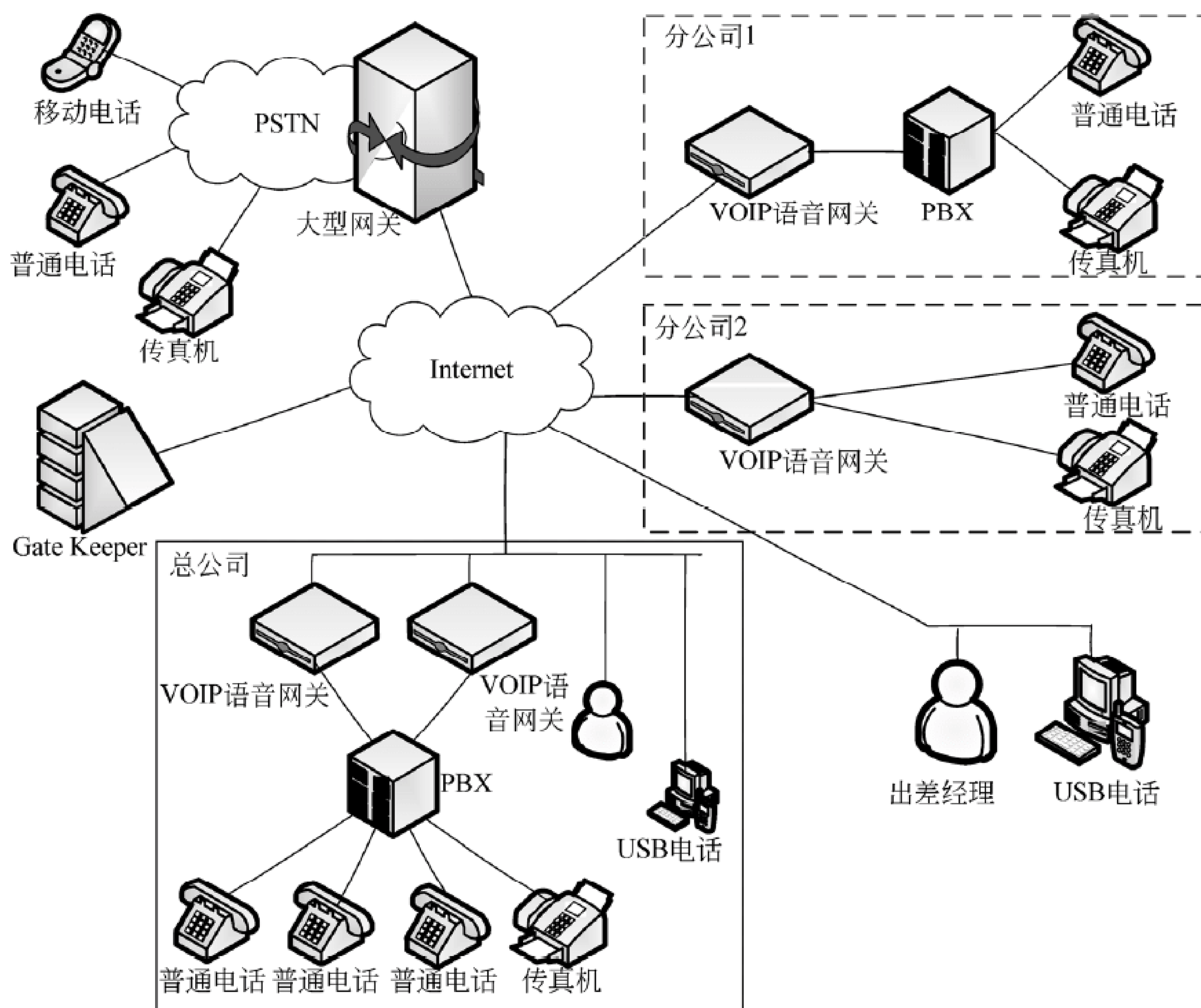


图 8.14　VoIP 的系统架构图

- 媒体网关器（Media Gateway）：主要扮演将语音信号转换成为 IP 封包的角色。
- 媒体网关控制器（Media Gateway Controller）：又称为 Gate Keeper 或 Call Server。主要负责管理信号传输与转换的工作。
- 语音服务器：主要提供电话不通、占线、忙线时的语音响应服务。
- 信号网关器（Signaling Gateway）：主要工作是在交换过程中进行相关控制，以决定通话建立与否，以及提供相关应用的增值服务。

4．VoIP的市场需求及成长空间

随着中国网络环境的不断改善，如带宽的增加，企业局域网的普遍建立，接入方式选择的增加等，网络通信系统的使用环境已经具备。同时语音的编解码技术、在 IP 网络上传输语音的技术也已经成熟，通话质量可以和传统电话相媲美。

如今，VoIP 电话（VOIP 电话）的市场需求正在爆炸性地增长。根据 Probe Research 的报告，全球由互联网传输的语音信号在 2000 年为 77 亿分钟，而这一数字在 2005 年达到了 57000 亿分钟。

增长的原因主要是：政府对电信网络管制的放松；服务供应商需要更快地通过新的服务来获得利润；VoIP 电话信号质量与可靠性的大幅度提高。由此可以看到，VoIP 电话具有巨大的市场发展潜力。

5．VoIP的技术及功能特点

VoIP 在应用中，主要有两个典型的特点：

（1）易于扩展，兼容性强

- IP 语音网关是基于模块化设计的多功能 IP 接入设备，具有良好的扩展性，可根据需要随时添加语音端口。
- 可以采用灵活的接入方式：支持 ADSL、DDN、ISDN，WaveLan 等，方便以后的分公司采用多种网络接入方式添加 IP 语音系统网。
- 兼容性好，互通性强。符合国际、国内相关标准，可以与同类产品互连互通。

（2）投资少，安装简单，易于维护

- 配置简单：本方案选用设备数量少且配置简单，整个网络所需投资少。
- 安装维护简单：无须专门的维护人员和额外的费用，可以实现远程的安装和调试。

8.2.3　Skype 与 P2P 技术

仅从 Skype 的功能上看，Skype 是一款提供语音对话、即时消息、文件传输和电话会议的即时通信工具，和 MSN、QQ 的区别不大。但是 Skype 的底层构架和关键技术与以往的即时通信软件有很大的不同，这正是 Skype 取得巨大成功的内在原因。而这一关键技术就是 P2P 技术。

关于 P2P 的基础知识，在本书的前几章节中已有详细的阐述，P2P 最本质的含义就是“对等、共享”，该技术最早是用于网络中对等结点之间的资源和信息共享的技术，如 BT、eMule 技术等。后来，Skype 在网络通话业务系统中灵活应用了该技术。由于冲击了传统通信领域，Skype 在引起很多争议的同时也使人耳目一新，可以说，Skype 发展和演化了

P2P 应用。

Skype 从网络结构上讲，是一种混合型的 P2P 网络结构，它结合了集中式和分布式的特点，在网络的边缘结点采用集中式的网络结构，而在超级结点之间采用分布式的网络结构，混合模式的 P2P 网络模型如图 8.15 所示。

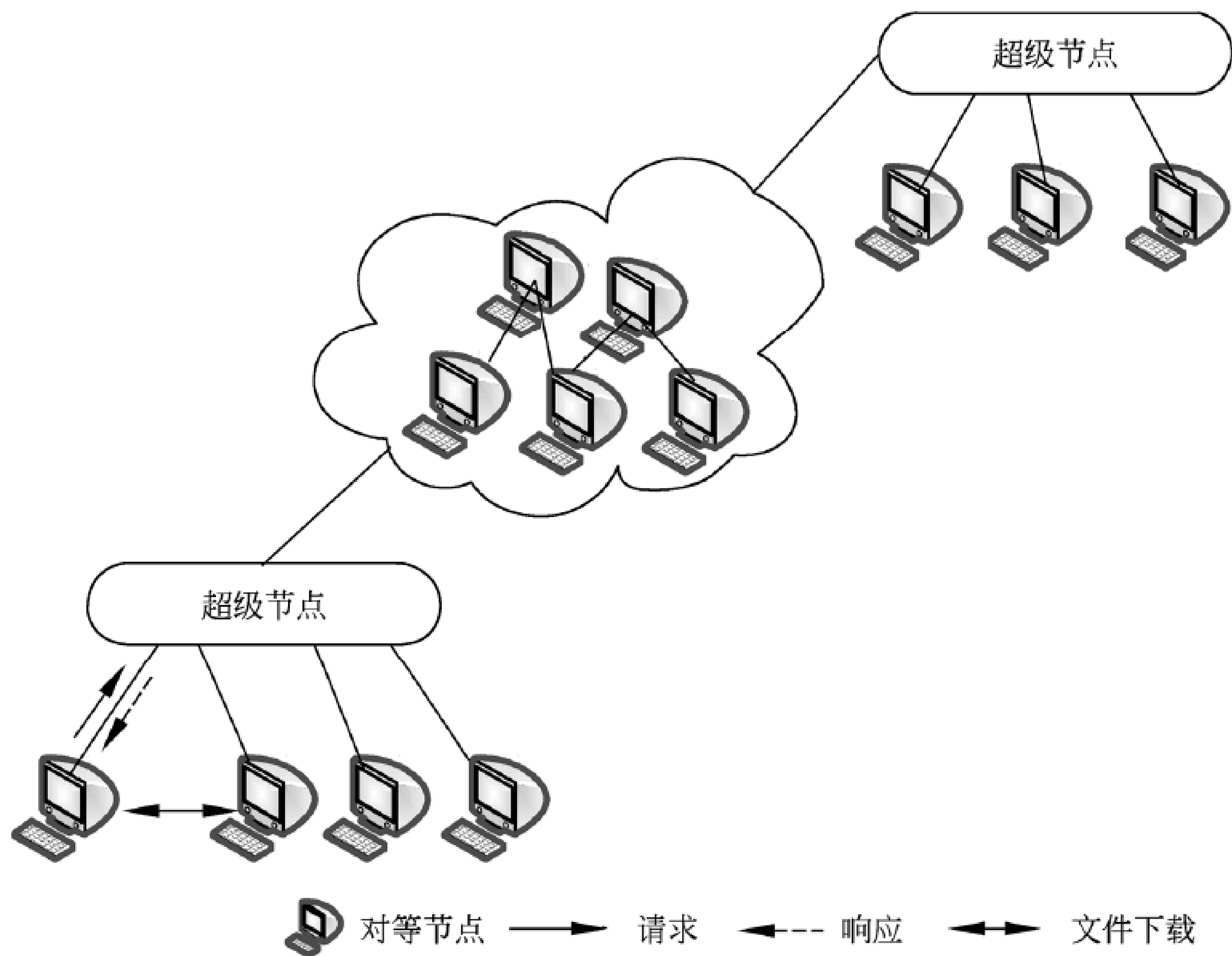


图 8.15　混合式的 P2P 网络模型

Skype 是基于 KaZaA 框架的对等网络应用系统。Skype 在运行的时候会在计算机上开启一个网络联机端口来监听其他 Skype 用户的联机呼叫；当其他计算机能顺利联机到这部计算机，Skype 称呼该用户为 Super node（如图 8.15 中所示的超级结点）。Super Node 在该 P2P 环境中的角色，即为提供其他无法被联机的用户之间的中继站，借用诸多 Super Nodes 的些许网络带宽，协助其他的 Skype 使用者之间能够顺利的互相联系。这种行为，在 P2P 环境中，算是相当常见的手法，也是点对点联机的精髓之一。Skype 是第一个将此种做法运用到网络语音通话与即时消息应用层面上的。

注意： KazaA 框架，就是一种混合式的 P2P 网络框架。在混合式的 P2P 网络结构中有超级结点和普通结点两种类型的客户端，普通结点以超级结点为中心形成集中式的网络结构，而所有的超级结点又以全分布式的结构构成整个网络，如图 8.16 所示。这种 P2P 网络，兼顾集中与分布，所以叫混合式的 P2P 网络，也叫 KazaA 框架。

在 Skype 网络中有注册服务器、普通结点和超级结点 3 种角色。注册服务器用来完成注册认证，普通结点仅通过运行 Skype 应用来进行语音通话和消息传递。超级结点同样是一组运行 Skype 应用的结点，只是超级结点必须具备一些特殊条件，即超级结点必须具有一个公网 IP；超级结点的处理能力、存储能力和带宽高于普通结点等。下面分别来讲解这 3 个角色的详细功能。

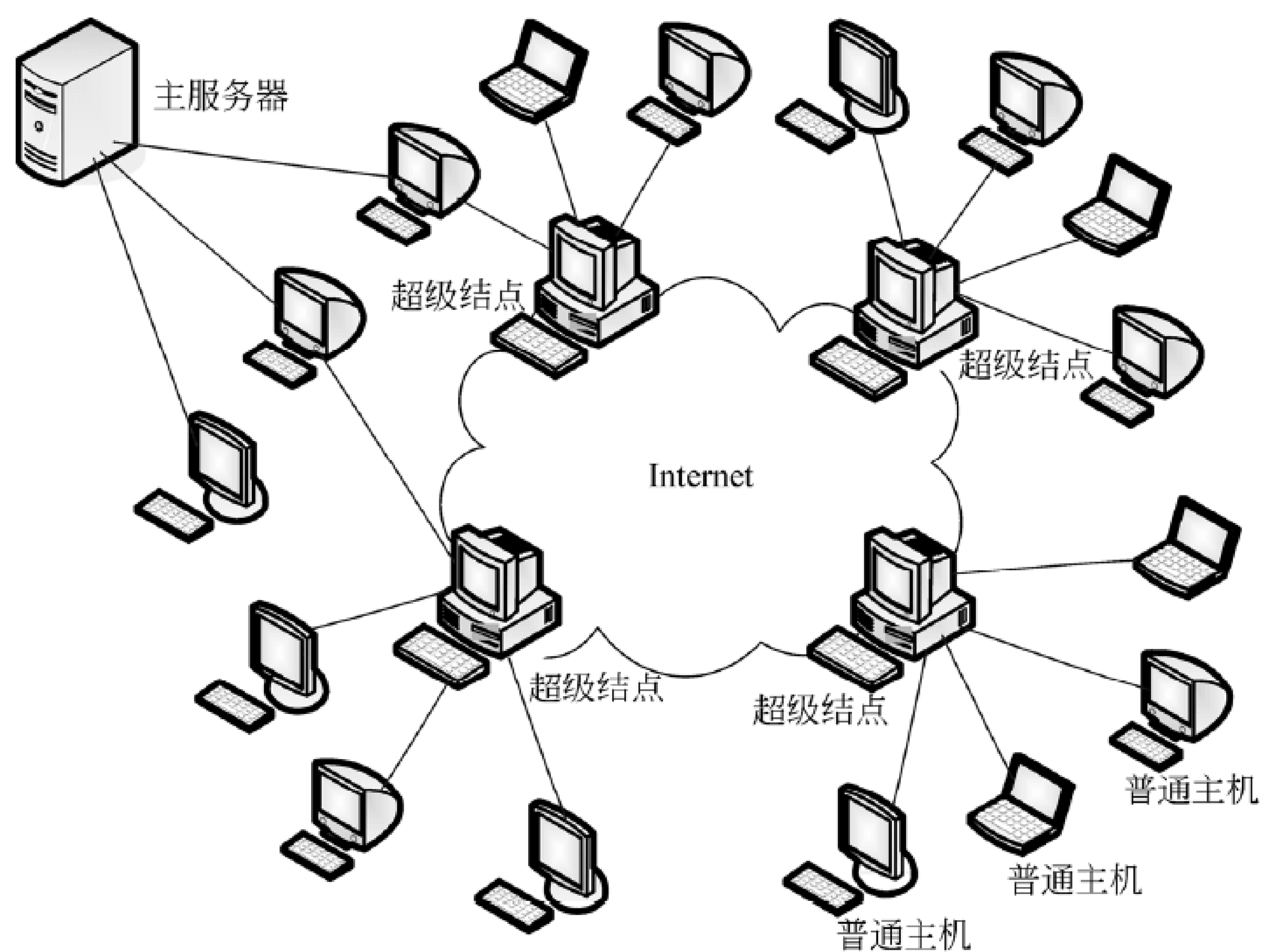


图 8.16　P2P 网络中的 KazaA 框架图

普通结点和超级结点都需要先通过登录服务器来加入 Skype 覆盖网，登录服务器不是 Skype 覆盖网的组成部分，是特殊的控制结点，这个控制结点就是所说的注册服务器。

注册服务器是 Skype 系统中唯一的有集中控制功能的服务器。它存储着所有 Skype 用户的用户名和密码信息。Skype 系统的所有用户的登录信息（主要包括用户名和密码）记录在登录服务器上；登录服务器验证用户的登录合法性，并保证用户名字符串在整个 Skype 覆盖网上的唯一性。

用户通过它进行注册认证，广播它的在线状态和好友信息，并能够检测用户是否位于防火墙或 NAT 后，同时判断防火墙的种类。另外，通过注册服务器 Skype 客户端软件，还可以获得更新的超级结点列表，用于后续的通信。

注意： 由于 Skype 覆盖网采用的是对等网络的分布式组织方式，因此，每个 Skype 结点必须建立和维护一定的路由信息和拓扑维护信息。

注册之后，Skype 客户端就不再需要注册服务器的参与，VoIP 接续和用户查找功能都由 Skype 普通结点和超级结点组成的 Skype 网络完成。普通结点就是可以进行普通的 VoIP 呼叫和即时消息的客户端。超级结点除了可以进行普通结点的通话和即时消息外，还可以当作普通结点的转接点。Skype 网络的整体架构如图 8.17 所示。

在 Skype 网络中，普通结点和超级结点都是由 Skype 客户端来充当的。每个 Skype 客户端运行时可以通过两种方式工作，一是作为普通结点，另一种是作为超级结点。

一个 Skype 客户端软件运行在何种状态下是由软件根据带宽和处理器性能自动选择的。任何一个客户端，只要它具有公共的 IP 地址、具有足够的 CPU，内存和带宽都可以成为超级结点，完成其他普通结点的转接工作。每一个 Skype 客户端软件都会维护一个超级结点的列表，如果一个超级结点连接不上，它会尝试与列表中的其他结点进行连接。

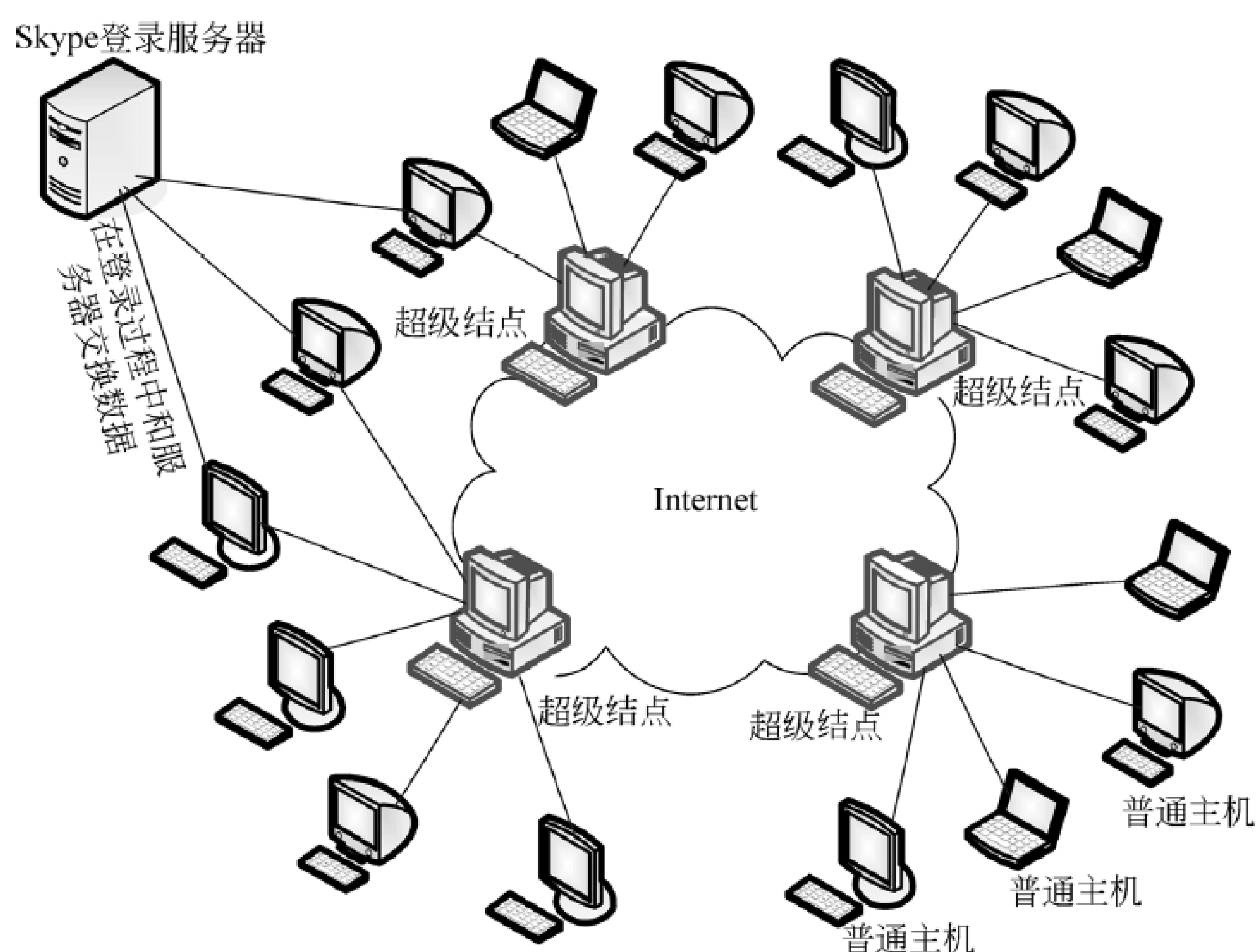


图 8.17　Skype 网络结构图

另外，为了保证 Skype 网络的可用性，除了普通用户的客户端可以作为超级结点，网络中还有一些固定的超级结点。这些结点可以作为 Skype 客户端首次运行时的默认超级结点来帮助客户端完成呼叫功能。也就是说一个普通的 Skype 客户端在运行时，除了在注册时需要与 Skype 的注册服务器通信进行用户口令的认证，其他时候不需要与任何集中的服务器进行通信，所有的呼叫信令和媒体数据流都由超级结点来完成。

注意： 呼叫信令，是由 ITU-T 定义的一组电信协议，描述了如何管理基于包网络上的视频、音频、数据和控制信息。

Skype 使用宽频带音频编码，保证在仅使用 32kb/s 带宽的条件下，实现高质量的语音通话。Skype 覆盖网中的控制信令使用 TCP 协议传输，多媒体传输使用 TCP 和 UDP 协议。控制信令和数据传输，使用不同的网络端口。

8.2.4　Skype 的基本功能

Skype 作为一个即时通信软件，除了拥有一般即时通信系统的基本功能以外，还有很多特有的功能和机制，尤其是作为 P2P 技术在即时通信方面的典型代表，Skype 还有很多其他的即时通信系统所不具备的在 P2P 技术方面的特性。

Skype 采用了最先进的 P2P 技术，可以提供超清晰的语音通话效果，使用端对端的加密技术，保证通信的安全可靠。除此以外，它还具备以下最基本的即时通信功能。

1．超清晰语音质量

免费多方通话，全球通用，采用“端对端”加密，极具保密性，跨平台使用，拨打普通电话使用起来超级简单方便，极强的穿透防火墙能力，可以与所有防火墙、NAT 和路由

器一起使用，且无需进行任何配置！

2．免费的多方通话

使用 Skype，还可以进行多达 5 人的清晰的多方会议呼叫，完全免费，而且操作简单，所有的通话都采用端对端加密，任何人无法截取信息。这不仅可以与好友们联络感情，还可以方便地进行商务会谈。

3．快速传送超大文件

文件传输功能同样采用了 P2P 的技术。文件在传输过程中进行了加密，安全性高。传输的文件尺寸大小无限制（可达若干 G），支持断点续传。文件传输可跨平台进行，即可以在不同的操作系统 Windows、Mac 和 Linux 平台间进行。

注意： 断点续传，指的是如果传输过程中因为网络原因或者意外中断，下次启动后可在原来的基础上继续传输。

4．无延迟即时消息

和普通的即时通信软件一样，也可以给好友发文本消息，速度更快。采用端对端加密，更保密、更安全。

5．全球通用

提供了全球搜索目录，可以根据不同的查询条件查询认识或者不认识的在 Skype 网友，并且，可以马上开始进行畅通无阻的语音聊天，让你的沟通更无障碍，效率更高。

6．跨平台使用

Skype 可以同时在 Windows，Mac OS X，Linux，和 Pocket PC 平台的 PDA 上使用，并且在不同的平台上有适合本平台的界面。语音聊天、发送即时消息、甚至传送文件都可以在不同的平台之间进行。你无须因为使用的不同的操作系统而无法与朋友们进行沟通。

8.2.5　Skype 的技术优势

Skype 之所以引起了不小的轰动，是因为它的互联网特性，即免费、开放和较好的业务质量。事实上，Skype 最大的意义在于，它开创了将 P2P 技术引入到语音通信的先河。也就是说，采用了网络中的所有结点都动态参与到路由、信息处理和带宽增强等工作中的机制，而不是单纯依靠服务器来完成这些工作，因此其管理成本大大降低，同时又保证了语音质量。

从具体技术的角度来看，Skype 的优势有下面几点。

1．穿透防火墙

大多数的 Voice-over-IP 应用程序不能穿透防火墙和 NAT（网络地址转换）。几乎所有宽带用户都处于防火墙和 NAT 之后，所以它们不能用 VoIP 应用程序。Skype 不是一个

典型的 VoIP 程序，它运用的是 P2P 技术，几乎可以在所有的防火墙或者 NAT 之后工作。

大多数此类软件的进入端口号都是指定的，Skype 没有指定进入的端口号，而是在安装程序的时候随机选择一个进入端口。如此能增强穿透网络地址转换（NAT），因为如果有若干个位于 NAT 之后的用户采用相同端口的话，则 NAT 会使得语音质量降低。在这种机制下，Skype 首先识别 NAT 和防火墙类型，然后通过动态的选择信令和媒体代理，从而轻松实现 NAT 和防火墙的穿越。

2．安全加密

Skype 采用了端对端的加密方式，保证信息的安全性。Skype 在信息（语音、即时消息、文件）发送之前进行加密，在接收到的时候进行解密，即使在数据传输过程中需要经过其他结点进行中转，也完全没有可能在中途被窃听。

Skype 采用了数字签名的方式，保证存储在 P2P 网络中的用户数据不被篡改。由于 Skype 使用的是 P2P 的技术，用户数据主要存储在 P2P 网络中，在用户进行搜索等操作的时候从公共网络中获取。如此必须保证存储在公共网络中的数据是可靠的和没有被篡改的。Skype 对公共目录中存储的和用户相关的数据都采用了数字签名，保证了数据无法被篡改。

3．非集中式全球搜索目录

大多数即时通信或者其他类似的带通信功能的软件都需要集中式的用户目录，通过该目录来建立终端用户之间的连接，以及用户和动态 IP 地址的映射关系。

IP 地址在每次用户登录的时候都有可能改变。大多即时通信软件通过集中式目录和日志来跟踪用户是否在线等。集中式目录在用户基数增加到百万以上之后，资源耗费会非常大。

Skype 技术，将这些资源耗费分散在了可利用的各个结点上，从而有更多的精力来开发前沿的技术和先进的功能。

全球搜索目录（GI）代表了另一种意义上的可扩展网络技术。全球搜索目录使用多层的网络结构，这种结构利用超级结点（Supernodes）来实现网络中的每个结点可以获取所有其他可利用结点的资源，并保证最小的延时。

4．最大可能的节省资源

Skype 对网络带宽的要求比同类产品低，Skype 在 33.6kbps 或者以上的 Modem 来拨号上网的情况下，也可以使用语音通话。Skype 可以根据双方的连接情况自动选择最佳的编码方式。语音通话的时候平均占用带宽大约是 3～16KB/S，实际占用带宽会根据对方的带宽情况、网络状况，以及 CPU 性能等有所不同。当空闲的时候大约只需要 0～0.5 KB/S 的带宽，主要是用来更新好友在线信息。具体的带宽情况可能会受许多因素的影响。

用 Skype 进行文件传输的时候，如果双方不能直接连接，则会通过其他用户的资源来进行中转。普通的同类软件在无法直连的时候一般是通过服务器中转，Skype 不利用服务器的资源，而是通过网络中的其他用户机器来进行中转。为了不过多占用做中转的用户资源，将速度限制在了 500K/S 以下。

Skype 是混合式的 P2P 网络模型，所以 Skype 在运行的时候，将很多工作下放给分散的网络结点去完成，大大地降低了中心服务器的负担，进而减少了维护和管理的成本。

5．跨平台

目前大多数 IM 软件都只能在 Windows 平台上运行。即使有单独开发的针对其他平台的版本，也往往是功能非常弱，例如只限于文本信息交换。Skype 目前有完全适用于 Windows 操作系统、Pocket PC、和 Mac OS 和 Linux 操作系统的版本，语音通话、文件交换等数据传输都可跨平台进行。Skype 所采用的底层技术保证了其可以很容易地移植到不同的终端设备上，更加适应终端设备和通信技术的发展。

Skype 采取开放的机制，在跨平台的同时，也鼓励互联网用户自己开发插件，目前此类开发如雨后春笋，在互联网上遍地开花。

6．使用简易、功能强大

Skype 具有迄今为止最优质的语音。现在很多 VoIP 和聊天工具都无法和 Skype 相媲美。Skype 与最优秀的声学科学家联手创造的独家拥有版权的软件，可以传递甚至高于固定电话质量的语音。用专业术语来说，传统的电话只能听到介于 300Hz 到 3000Hz 频率的语音。Skype 可以听到所有频率的语音，从最低沉的到最尖锐的。

Skype 强大的功能还有以下几点。

- 很高的呼叫成功率：没有其他任何一个互联网技术系统可以有和 Skype 一样高的呼叫连通率。
- 使用简单：现在 VoIP 应用程序配置很困难，不熟悉网络和计算技术的用户几乎无法使用。Skype 无论在软件还是硬件方面，用户都无须做任何收工的设置，通常只要注册一个账户就可以立即登录，开始语音通话了。
- 多方语音通话：Skype 在同类软件中首先提供了免费的多方语音通话，采用混音的方式，操作简便、音质良好，且尽可能地节省网络和机器资源。
- 快速路由机制：Skype 采用了全球索引（Global Index）技术提供快速路由，其用户路由信息分布式存储于网络结点中。

7．结合互联网特点的语音编解码算法

Skype 通过与 Global IP Sound 公司合作，引入语音质量增强软件，专门针对互联网的特点，从而降低了业务对带宽的要求，并且保证了 Skype 的通信质量。

在 Skype 以上的这些特性中，其中第 1 条保证了通信无障碍，无论终端处于何种网络条件，都不会影响用户使用 Skype 提供的业务。第 2 条保证了 Skype 的安全，第 3 条保证了 Skype 在市场竞争中的巨大优势，而第 4 条则给了 Skype 更强大的生命力，使其更加灵活，具有更高的可扩展性和更广的应用范围。第 5 条则保证了 Skype 较好的业务服务质量，也使得 Skype 可以轻松面对挑战，始终保持在技术和应用上的领先地位。

8.2.6　Skype 的与众不同

上文中已经很详细讲解了 Skype 的很多特性和功能，但在当前的互联网上，即时通信系统和语音视频聊天软件层出不穷，它们的特点如何？与它们相比 Skype 与众不同的地方在哪里呢？

1．Skype与即时通信系统的不同

就典型的即时通信系统 MSN 而言，MSN 的视频效果最好，还可传文件。但双方只要有一方在内网，便只能传视频及文件，不能传语音。由于 MSN 语音传送的特殊性，除了用 UPNP 路由外，即使进行端口映射也不能实现语音通信。显然我们在自组的局域网中是无法对外实现语音聊天的。

还有另一款 Net Meeting 系统，安全性好，可直接输入对方 IP 进行呼叫，无须在服务器进行注册，但双方必须至少有一方在公网，由内网呼公网，可实现视频、语音通信。另外，还可使用白板，但视频效果要差于 MSN，语音效果也要差些。如果双方均在内网，便不能使用。

至于其他的即时通信软件几乎都存在着类似的问题，要不是不能进行 NAT 的穿透，就是无法保证安全，而且即便能够传输语音，其音质、效果也较差。

注意：白板，一种类似画图的软件，双方均可进行绘画，并立即在对方电脑上显示出来，双方可共同进行绘图交流。

之所以说 Skype 是最好的纯语音软件，是因为 Skype 可穿过双方 NAT 及防火墙，无须端口映射。双方均在内网，也能实现语音通信，而且语音效果非常好，应比 MSN 语音还要好，实际使用中，与打电话一样，声音非常清晰。

2．Skype与其他VoIP系统的不同

Skype 不同于传统 VoIP 系统的主要方面之一是可以无缝地穿越防火墙和 NAT（网络地址转换）设备，用户无须进行任何配置。

在现实网络中，由于 IPv4 的地址资源问题和安全问题，防火墙和 NAT 设备的使用非常频繁，一般大多数 VoIP 客户端都是以 RTP 协议进行媒体传输的，这也严重影响了 VoIP 的可用性。

因为，防火墙设备可以针对 TCP 和 UDP 协议以及端口进行限制。很多的企业防火墙为了安全起见会禁止未知的 UDP 端口，而一般的 VoIP 媒体流都是以 UDP（一般都是通过 RTP）形式传输的。也就是说，如果一个企业想禁止 VoIP 语音通信，只要禁止所有未知的 UDP 端口（一般是端口号 10000 以上）就可以了。

除了防火墙可以人为地影响 VoIP 系统，NAT 设备从设计原理上就会导致 VoIP 的不可用。在 H.323 或 SIP 等 VoIP 协议中，主叫和被叫客户端会通过呼叫信令协商会话中所采用的 IP 地址和端口等信息，当会话建立起来后，媒体流会使用协商好的 IP 地址和端口等信息进行传输。而如果客户端在 NAT 设备之后，NAT 设备会对 IP 包头的地址进行转换，但不识别应用层 VoIP 信令内的 IP 地址，如果 VoIP 系统不进行相应的处理，呼叫将不能正确建立。

Skype 系统在防火墙和 NAT 的处理上做的非常智能，在不要用户设置的情况下可以自动做出识别，完成在防火墙和 NAT 后的呼叫接续。

首先，针对防火墙，Skype 可以做到信令和媒体流即可以在 UDP 上传输，也可以在 TCP 上传输。对于信令，Skype 通常使用 TCP 进行传输，它可以使用 TCP 端口 80（HTTP 端口）和 443（HTTPS 端口），一般防火墙不会禁止这两个端口。对于媒体流，Skype 将

首选 UDP 进行传输（UDP 没有差错重传机制适合媒体流的传输），如果不成功，它也会使用 TCP 进行媒体流的传输。在针对 NAT 的处理上，它使用了一种类似于 STUN（RFC3489）的协议来识别 NAT，并进行相应的处理。

🔔注意：关于 NAT 穿透的详细知识，请参阅本书第 5 章的相关内容，如图 8.18 所示的就是 Skype 中 NAT 穿透的模型，读者对照第 5 章所讲内容进行理解。

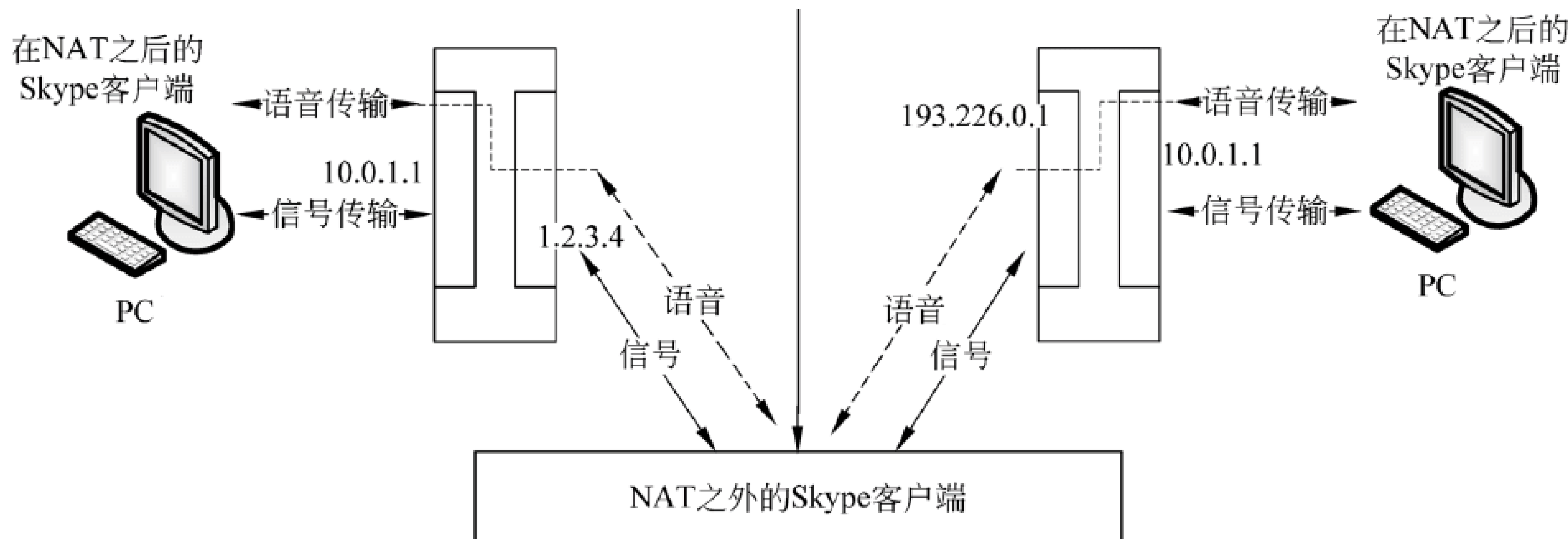


图 8.18　Skype 中的 NAT 穿透模型

Skype 的呼叫接续是使用自定义的信令协议，而没有使用 SIP、H.323 等 VoIP 信令，它的所有通信协议都是加密传输的。它的呼叫接续以尽力连接为前提。

🔔注意：SIP 是一个应用层的信令控制协议。用于创建、修改和释放一个或多个参与者的会话，而 H.323 是 ITU 多媒体通信系列标准 H.32x 的一部分，该系列标准使得在现有通信网络上进行视频会议成为可能。其中，H.322 是在有服务质量保证的 LAN 上进行多媒体通信的标准。这些都是电信方面的知识，不明白的地方请自行查阅相关资料。

根据 Skype 客户端所处的位置，可以考虑 3 种呼叫场景，即主叫被叫有公共的 IP 地址，主叫或被叫有一个在防火墙和 NAT 后面，主叫和被叫都在防火墙或 NAT 后面。当 Skype 主叫端发起呼叫时，主叫会和与它相连的超级结点建立 TCP 连接，并通过这个 TCP 连接进行呼叫信令的传输。与此类似被叫端也会和与它相连的超级结点建立 TCP 连接，用于信令的传输，如图 8.19 所示。

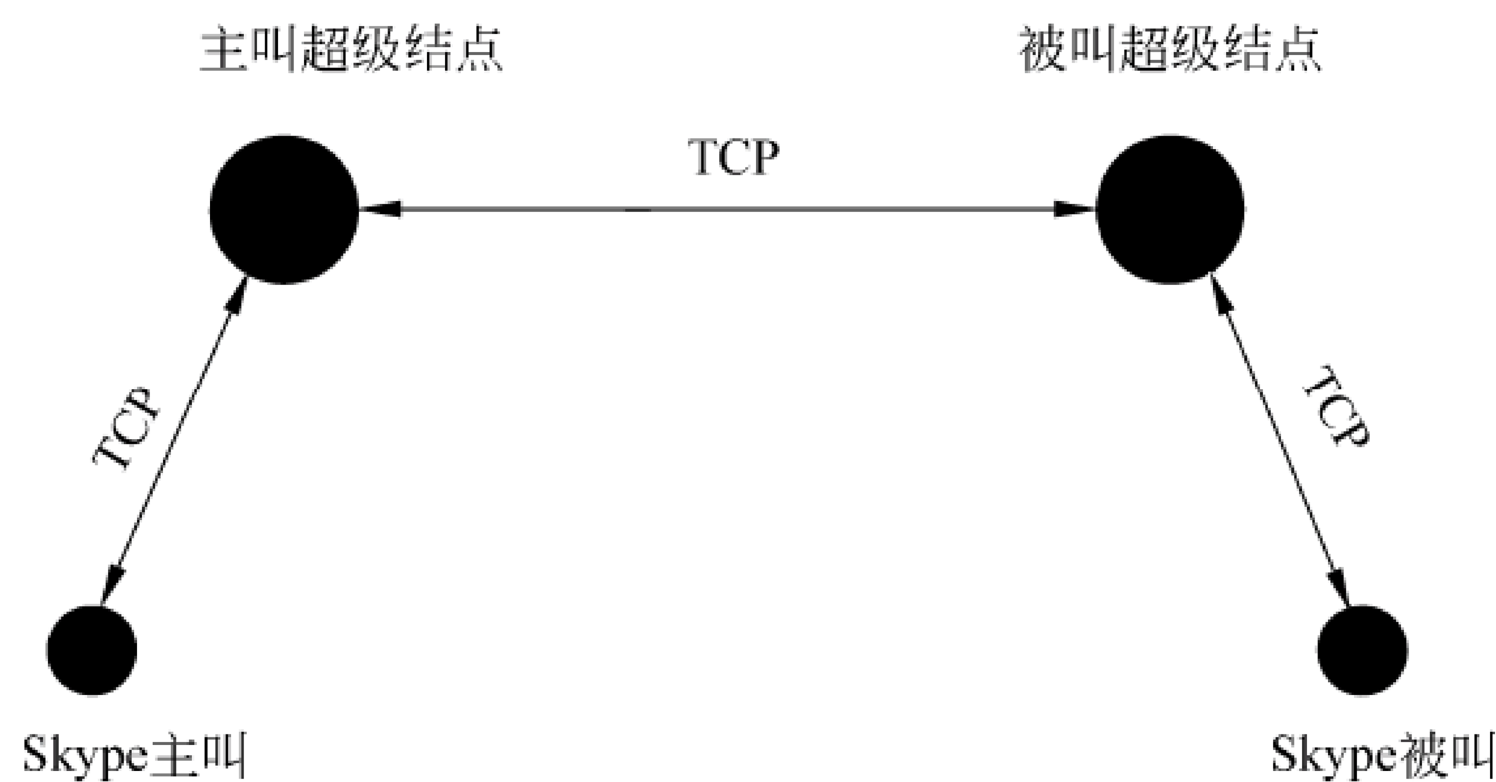


图 8.19　Skype 呼叫的信令连接

当通话建立以后，对于媒体流的传输，将尽可能地选用 UDP 进行传输。与呼叫信令的工作方式类似。

- 如果 Skype 主叫和被叫都在公网上，则主叫和被叫之间直接通过 UDP 进行通信。
- 如果主叫或被叫在 NAT 后面，则它们通过超级结点进行媒体转发，也使用 UDP 进行通信。
- 如果主叫和被叫都在 UDP 被禁止的防火墙和 NAT 后面，那么它们将使用 TCP 连接到超级结点，并由超级结点进行媒体转发。

另外，Skype 并没有实现完全的静音抑制，即使在主叫和被叫都没有说话时，仍然有媒体数据进行通信。这样做的目的主要是，在使用 UDP 传输时保证 NAT 端口的激活，而在使用 TCP 传输时保持它的拥塞检测窗口尺寸。

Skype 不同于传统 VoIP 系统的另一个主要方面在于被叫定位。在传统的 VoIP 系统中（如 SIP，H.323），每一个用户注册时都会将自己的 IP 地址告诉注册服务器，其他用户可以根据用户名来查找到他们将要呼叫的用户的地址。

Skype 是分布式的网络结构，没有中心的数据库存储所有注册用户的地址。客户端主要是通过超级结点发现另一个用户的 IP 地址，当主叫客户端发起一个会话时，它首先会连接到一个临近的超级结点，该超级结点会根据要查找的用户发给主叫客户端一些结点信息（一般第一次超级结点会发送 4 个结点信息）。Skype 客户端会与这些结点连接查找用户，如果找不到，它会通知超级结点，超级结点会再给它发一些结点信息（第 2 次一般为 8 个）让它继续查找。一般来讲，Skype 客户端可以在与 8 个结点连接后查找到目标用户。

以上描述的 Skype 与众不同之外，可以图 8.20 所示的对比直观地表示出来。

	skype	MSN, Yahoo Messenger ICQ, AIM	其它标准 VoIP客户端
与防火墙/NAT 设置一起使用**无需进行任何配置**	✓	✗	✗
对同一应用的用户进行**不受限的免费呼叫**	✓	✓	有时
音质	★★★ 比普通电话好	★ 比普通电话差	★ 比普通电话差
安全加密通信	✓	✗	✗
100% 免费	✓	✗	有时

图 8.20　Skype 与 VoIP 及其他即时通信系统的对比

图 8.20 中，列出了 Skype 与 VoIP 和其他即时通信系统客户端，在配置、安全性、音质等方面的对比情况。通过这个对比可以清楚地发现，Skype 几乎具备了所有的优点，自然就是与众不同了。

8.3　Skype 的技术分析

Skype 是由 KazaA 于 2003 年发明的基于 P2P 技术的即时通信系统，用户通过 Skype

可以在互联网上进行语音和文本的传输，兼具即时通信和 VoIP 的双重功能。Skype 与同类产品相比具有领先的技术优势。但 Skype 的通信协议是不公开的，而且其通信内容也是加密的，这就为研究 Skype 技术带来一定的困难。本节从研究和探索的角度对 Skype 的相关技术进行分析。

注意：由于 Skype 的私密性质，所以本文无法从更底层角度去阐述 Skype 的技术实现细节，以下所讲的 Skype 相关技术只是一种探究性的、引导性的讲解，更深层的知识需要读者自行研究。

8.3.1　Skype 的网络结构

利用 Skype 可以实现文本消息的即时交互和语音传输，那么 Skype 在整个工作流程中是如何实现这一过程的呢？下面就讲一下 Skype 的通信原理。

与常规的电信业务网络不同的是，Skype 的网络中除了注册服务器，没有其他任何集中的服务器，只是将用户结点分为普通结点和超级结点。Skype 的系统连接结构如图 8.21 所示。

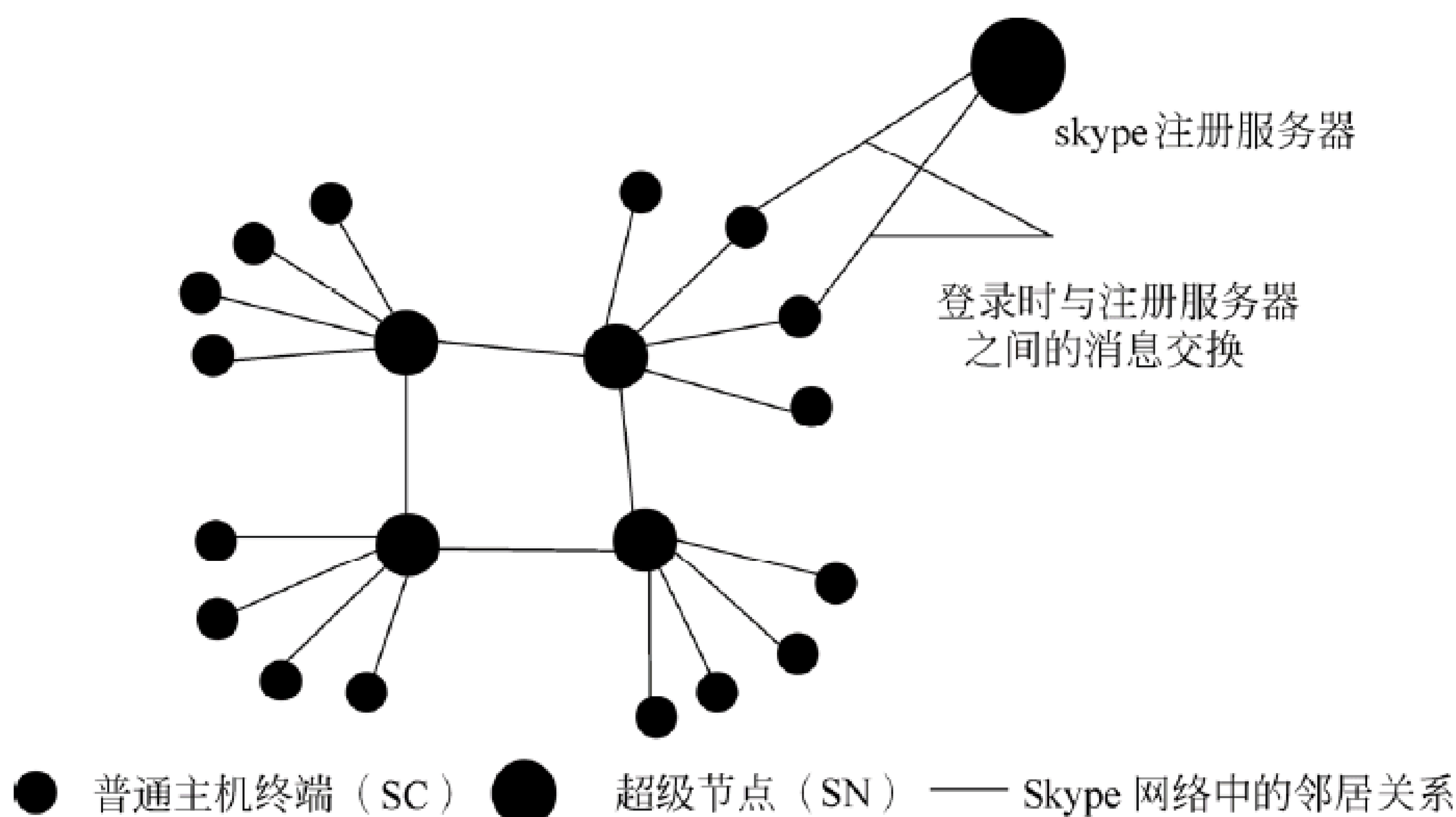


图 8.21　Skype 系统的连接结构图

注册服务器是 Skype 唯一需要维护的设备，它主要负责完成客户端的注册，存储并管理用户名和密码信息，当用户登录系统时，对用户进行身份认证。注册服务器还需要检验并保证用户名的全球唯一性。

Skype 的客户端结点，分为普通结点和超级结点两种类型，普通结点即普通主机终端，只需要下载了 Skype 的应用程序，安装使用后就具有提供语音呼叫和文本消息传送的能力。

超级结点实际上是满足某些要求的普通结点，这些要求包括：具有公网 IP 地址、具有较强的 CPU 计算能力、足够大的存储空间和流畅的网络带宽。也就是说，在安装和开启了 Skype 应用的前提下，任何符合条件的主机终端都可以成为超级结点。

8.3.2　Skype 的基本构件

Skype 的主要构件是指实现 Skype 一系列功能模块的组件，这些组件有的是可以直观

看到的，有的是需要通过推理分析出来的。从 Skype 启动到 Skype 完成通信，这一过程基本会经历这几个步骤。

- Skype 客户端（SC）在指定端口侦听呼叫信息；
- 缓存结点信息，以维护更新其他 Skype 结点的列表；
- 宽带编解码；
- 维护好友列表；
- 加密发送消息，并确定自身是否处于 NAT 和防火墙后。

从以上这几个基本步骤里，可抽取出这样几个处理过程：处理端口信息、需要对结点信息进行缓存、需要编码解码、要处理好友列表、对消息进行加密和 NAT 穿透。每个过程对应一个模块的话，Skype 的主要构件也就一目了然了。

1．端口

Skype 客户（SC）打开一个 TCP 和一个 UDP 侦听端口，在连接对话框中配置。客户端在安装时随机地选择配置的端口号。此外，客户也同时打开 80 号端口（HTTP 端口）和 433 号端口（HTTPS 端口）作为 TCP 侦听端口。

2．主机缓存（HC）

主机缓存（Host Cache，简称 HC）是记录超级结点 IP 地址和端口对的列表，客户端有规律的建立和更新该列表。它是 Skype 运行中非常关键的一部分。HC 中至少存在一个有效的信息，这个信息描述了一个在线 Skype 超级结点的 IP 地址和端口号。Skype 客户将 HC 存储 在 Windows 的注册表中。对于 Skype 而言， 类似于这样的主机和结点的缓存已不是新技术。在 Chord 协议中，就有一个 finger 表，类似于主机缓存的功能，可以用来快速查找结点。

注意： Chord 协议请参阅本书 P2P 网络体系结构部分的内容。

3．编码解码（Codecs）

Skype 白皮书提示 Skype 使用了 3 种音频编码，分别为 iLBC、iSAC 和第 3 个未知的编码方式。iLBC 和 iSAC 是由 GlobalIPSound 公司开发的。Skype 所采用的音频编码范围为 50～8000Hz，这个频率范围是宽带编码解码的典型特性，也可以称之为宽频带编码技术。

4．好友列表

Skype 将好友列表存储在 Windows 操作系统的注册表中，存储的内容进行了数字签名和加密。因此，好友列表仅存储于用户的计算机中，而不存储于中央服务器上。这一点与其他的即时通信系统不同，这也解释了为什么当用户在不同的计算机上登录 Skype 时，会发现其好友列表不同或需要重建。

注意： Skype 中会将好友信息存储在本地主机中，这个特性是基于安全性考虑的，如 QQ 等则不同，QQ 不管你从哪个地方登录，你的好友列表信息总是从服务器获取，好友列表信息不需进行重建。

5．加密

在 Skype 系统中，它的加密方式是使用 AES 方式对数据进行加密的。Skype 使用 256 位的密钥，即总共 1.1×107 个不同密钥。这样，可以保证每一个使用 Skype 进行的会话，可以使用不同的密钥进行加密。对于 AES 密钥的传输，Skype 使用 1536 或者 2048 位密钥的 RSA 加密进行传输。用户公钥在注册时由 Skype 服务器验证。

6．网络地址转换和防火墙

Skype 客户端，使用不同的 STUN 和 TURN 协议来确定它所处在 NAT 和防火墙的类型，所得到的信息存储在 Windows 注册表中，Skype 客户端会定期地更新这些信息。

8.3.3　Skype 的协议分析方法

虽然 Skype 的协议是私有的，但这并不妨碍对 Skype 的协议进行研究分析，哥伦比亚大学的 Baset 和 Schulzrinne 就完全在实验的基础上对 Skype 的通信机制进行了分析，分析的结论也准确地验证了 Skype 的一些特性。

下面就简要地介绍一下分析 Skype 的一些简要方法。

1．观察

现在要分析的对象是 Skype，分析的目的是研究它的协议，那么第一步要做的就是观察，观察是多方面的。

- 观察 Skype 的外观，通过外观的基本设置、界面布局、功能选项等了解 Skype 的基本使用方法、基本设置和操作方法。
- 观察 Skype 的运行过程，这一观察过程就要实际的去操作和使用 Skype 了，如使用 Skype 进行登录、添加好友、查找好友、通话呼叫、发送即时消息等，根据不同的操作方法，观察 Skype 的反应和操作的结果。
- 观察 Skype 的数据通信情况，这就需要借助一些第三方的软件来完成，比如最基本的抓包软件 WireShark 等，启动 Skype 的同时，也启动此抓包工具，将 Skype 的每一个通信过程的信息、发送的数据包、应答的数据包等都抓取下来，留待进一步的分析。
- 还有就是进行横向和纵向的比较观察，比如，在不同的主机上安装 Skype 客户端并观察相互之间的通信，在私有网络和公有网络上分别安装 Skype 进行对比观察，在不同的时段、不同的网络带宽下观察 Skype 的通信情况等。

要研究 Skype 的通信协议，观察是非常重要的一步，而且这些观察需要多次反复地进行。

2．实验

实验，就是在观察的基础上对自己的观察结果和设想进行验证的过程，这一过程是整个协议分析过程中最重要的一步，所有的结论都源于实验的结果。

比如：Skype 在安装后第一次运行的时候，它会发送一个 HTTP1.1 GET 方法的请求信

息到 Skype 服务器。此请求的第一行中包含了关键字 installed……对于这样一个结论，就需要通过实验来验证。首先要保证 Skype 是安装后的第一次运行，其次观察 Skype 发向服务器的通信数据包，是否包含着以上信息，同时还要进行对比实验，当 Skype 不是第一次运行的时候，它发送的数据又是什么？这样的实验要保证在不同的条件下做 5 次以下，如果所有的实验结果都是上面的结论，那么基本上这个结论就是确定的了。

还有些实验是验证性，比如我想验证一点，由公网用户查询私有网络里的用户时，缓冲查询结果的中间结点是否存在？针对这个推论可进行如下验证。

- 用户 A 是处在限制端口的 NAT 和限制 UDP 的防火墙后的客户端，它注册进了 Skype 网络。
- 用户 B 是公网的客户端，由用户 B 查找用户 A，发现查找的过程耗时 6～8 秒；
- 然后将 B 端的客户端卸载，清除 Windows 注册表中的 Skype 相关信息，之后重新安装客户端；
- 再次由用户 B 查找用户 A，这次查找过程耗时 3～4 秒，这样的试验在不同时间重复进行了 4 次，获得的结果都相似。

为什么会有这样的区别呢？既然已将 B 的客户端卸载并清除了注册表信息，说明在 B 的主机上不可以缓存 A 的任何信息。由此可以推断，之所以查询耗时会减少，一定是有某一个中间结点缓存了 B 对 A 的查询结果，所以在执行第二次查询的时候，直接从缓存里读取，用时自然也就减少了。

因此，从以上的试验，可以推断出客户端的查找结果缓冲在了某个中间结点中。

以上就是关于 Skype 协议分析中，关于实验的一些方法探讨，在实际的分析过程中，对每一个观察结论、的推导的假设，都要经过实验去验证，实验的方法和思路需要读者在不断研究中去积累和探索。

3．分析

分析，就是对观察结果和实验数据进行分析，从而形成论断，这个过程需要知识的积累、敏锐的直觉，也需要一点点的灵感。

4．总结

总结，就是将实验论断和结果，进行总结，在进一步实验验证的基础上得出实验报告，形成实验文档。

这里只是简单地介绍一下 Skype 协议分析的基本方法，对任何其他的陌生协议而言也大致是这种分析思路，具体的分析过程远比这些要复杂。下文所讲的 Skype 协议的一些内容就不再讲解是如何分析得到的，具体的分析过程建议读者自己去操作、去研究。

8.3.4 Skype 的通信流程分析

Skype 的通信流程分为：启动、注册（认证）、查找用户、呼叫和释放的过程。其中注册流程只是在用户初次安装了 Skype 的客户端软件后进行注册，后期使用的过程中该步骤就变成认证过程。

1．启动流程

Skype 的用户终端启动时，采用 HTTP 协议连接到注册服务器，根据用户使用 Skype 的情况，连接到注册服务器的时候有两种方式。

- 第一次安装 Skype 后，注册到服务器。
- 使用 Skype 时，启动后注册到服务器。

根据这两种不同的启动方式，Skype 客户端会向服务器发送不同的请示，当 SC（Skype Clinet，Skype 客户端，后文所说的 SC 都表示的是 Skype 客户端的意思）安装后第一次运行的时候，它发送一个 HTTP1.1 GET 方法的请求信息到 Skype 服务器。此请求的第一行中包含了关键字 installed。请求示意图如图 8.22 所示。图中标示的 LC，是 Login Server 的意思，就是 Skype 的注册服务器。

如果是用户在使用时启动 Skype，则在随后的启动中，SC 仅发送一个 HTTP1.1 GET 方法的请求信息到 Skype 服务器，来检查有没有 Skype 的新版本，此请求的第一行中包括关键字 get latestversion。这一请求过程如图 8.23 所示。

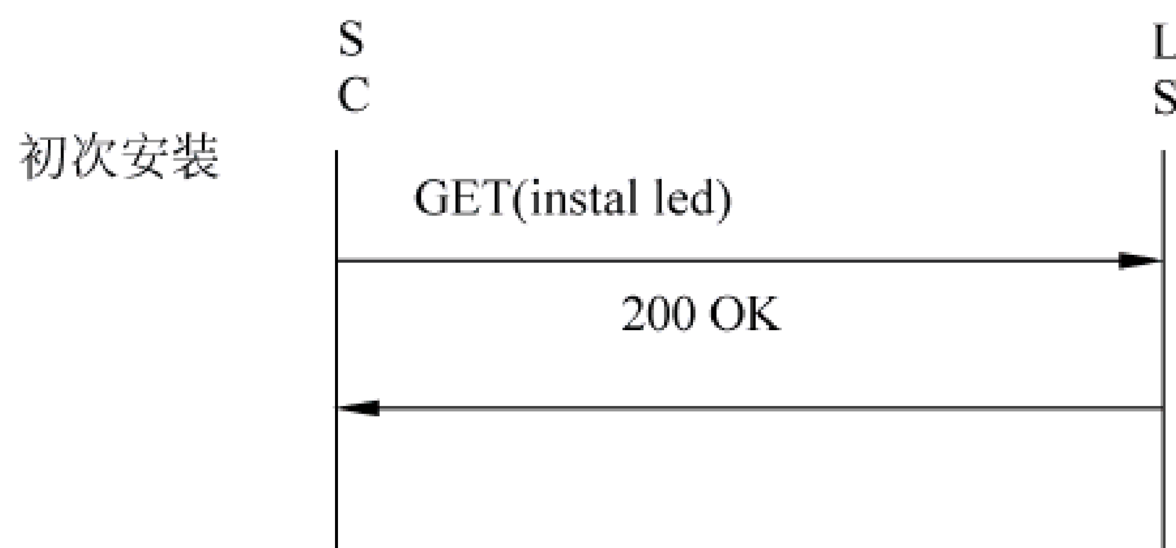


图 8.22　初次安装 Skype 时的启动流程

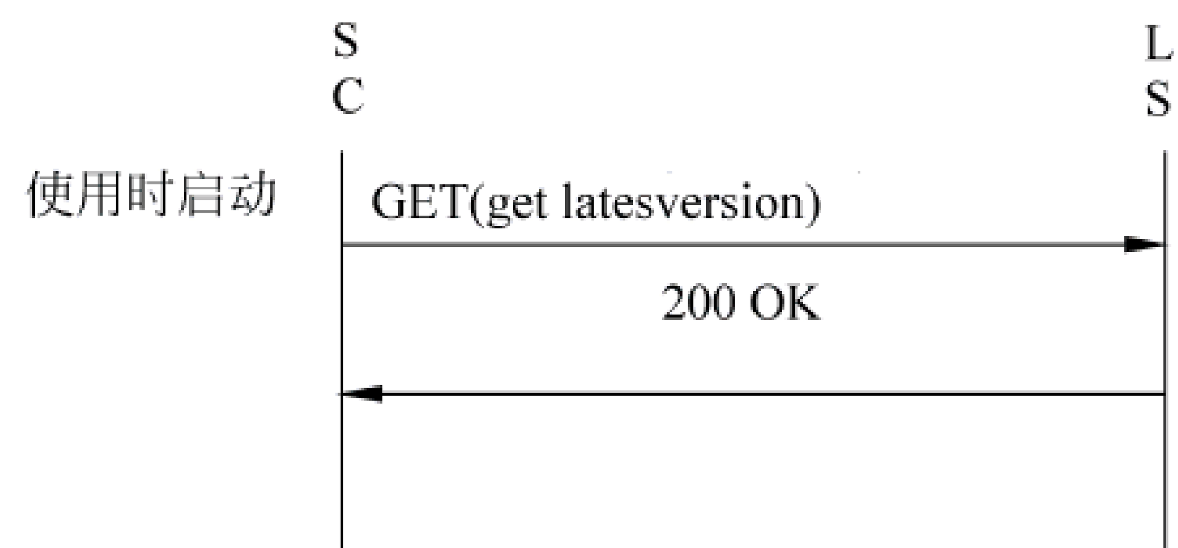


图 8.23　用户使用 Skype 时的启动流程

2．注册和认证的流程

Skype 启动以后，就进入一个注册（认证）的流程，这个流程是 Skype 所有流程中最复杂的一个。

用户启动 Skype 后，首先需要连接到超级结点，通过超级结点向注册服务器发送身份认证信息，注册服务器验证用户名和密码的合法性，然后向其他对等结点及其好友发送在线信息，同时还需要判断该终端所在网络的 NAT 和防火墙类型。

如果该终端先前默认的超级结点已不可用，则还要查找具有公网地址的 Skype 结点来作为该终端的超级结点，从而维持该终端与 Skype 网络的连接。

Skype 的注册和认证过程，需要完成以下几个流程：

（1）执行注册过程

在每个 Skype 的客户端，为防止其超级结点不可用，在用户安装 Skype 之初，就会在客户端就建立一个可选连接的结点列表，这个列表存储在主机缓存（HC）里，列表的内容包含至少 7 个 IP 地址和端口组的信息。在 Skype 执行注册过程的时候，会执行以下几个步骤：

- SC 首先发送一个 UDP 数据包，到 HC 中存储的结点列表所指的机器。
- 如果在大概 5 秒钟后没有响应，SC 尝试和该条目所指机器建立一个 TCP 连接，尝

试连接该条目所指的 IP 地址和 80 端口（HTTP 端口）。

- 如果仍然不成功，它尝试联系到 HC 中的 IP 地址和端口 443（HTTPS 端口），SC 那时等待大概 6 秒钟。
- 在报告注册失败后将重复整个过程 4 次以上。

SC，则连接到 Skype 网络必须和超级结点建立一个 TCP 连接。如果它不能连接到超级结点，它将报告注册失败。

注意：大多数防火墙的配置，允许端口 80 和端口 443 可以进行 TCP 数据流的发送。如果 SC 处在防火墙后，则防火墙可以阻断 UDP 流，但却可以选择性地允许 TCP 数据通过。利用这个因素，在注册时，Skype 客户端与一个公有 IP 地址和端口号为 80 或 433 的 Skype 结点建立一个 TCP 连接，其成功的几率就会增加。

（2）注册服务器

在客户端与超级结点连接之后，客户端必须通过注册服务器验证用户名和密码。注册服务器是 Skype 网络中的唯一中心元件，它存储了所有 Skype 用户的用户名和密码，并确定所有的 Skype 用户名是唯一的。

（3）连接到超级结点

在第一次安装程序并完成注册之后，用户的主机缓存（Host Cache）被初始化。在第一次注册过程中，HC 总是被初始化为 7 组以上的 IP 地址和端口号，这些地址和端口组所代表的便是初始的超级结点。客户端在第一次连接注册服务器时就是与 7 组中的一组建立 TCP 连接。这个 TCP 连接建立的过程，就是上文所讲的执行注册的过程。

（4）确定 NAT 和防火墙

客户端能够在注册的过程中确定它是否处在 NAT 和防火墙的后面，通常有两种方式来确定这些信息。

一种方式就是客户端使用 STUN 协议与超级结点交换信息来确定；客户端使用不同的 STUN 协议来确定 NAT 和防火墙的类型，从而判断终端所处私有网络的 NAT 和防火墙类型。

另一种方式是在注册的过程中客户端与超级结点建立 TCP 连接，然后发送数据到一些结点，再接收回复。

一旦确定 NAT 和防火墙的类型，客户端将这些信息保存到 Windows 的注册表中。客户端软件还采用定期刷新机制来定时更新，保证任何时候都能穿越 NAT 和防火墙。

（5）向好友发送在线信息

由于 Skype 采用路由缓存机制，即用户查找其好友的过程中会在中间的超级结点缓存其路由信息（缓存 72 小时），因此用户登录后，其状态信息可以通过其超级结点通知到好友终端，并将好友的状态返回给用户。一旦缓存超时，需要通过其他超级结点查找用户路由，这充分体现了 Skype 的用户路由信息动态分布式存储的特点。

3．用户查找

Skype 采用了一种称做全球索引（Global Index）的技术来查找用户，通过这种查找方式，在 72 小时内登录过的用户，无论是处在公网还是私有网络中都能找到。

该技术结合前面所述的分层网络，超级结点之间采用全分布式的连接，每个超级结点

具有最小时延前提下所有可用的用户和资源的全部信息。具体来说，Skype 采用了下面两种机制来保证顺利完成用户的查找。

- 启动后向所有列表中的用户发送其上线信息，其他用户响应各自的信息。
- 在中间结点缓存查找到的用户信息。

客户端有一个查找对话框，在输入了用户 ID 后单击“查找”按钮，客户端就开始查找指定用户。

对于有公网地址的客户端，其查找用户的过程如下。

- 它发送 TCP 数据包给它的超级结点，收到这个数据包的查询信息后，超级结点给它回复 4 个其他普通结点的 IP 地址和端口号，用于进一步查询。
- 与超级结点交换数据结束后，客户端接着发送 UDP 数据包给超级结点回复的那 4 个结点。
- 客户端发 UDP 包给这些结点，如果不能找到需要的用户，客户端就会通过 TCP 告知超级结点，没有找到目的结点。
- 超级结点收到客户端的第二次查询请求后，接着就会返回给客户端 8 个不同的结点，然后客户端再向这 8 个结点发 UDP 包，进行确认。

这样的过程持续到客户端查找到需要的用户或者确定用户不存在。返回用户的信息或是返回查询失败的信息。

对于那些位于私网内的受限客户端，它们的查询方式如下。

- 客户端通过 TCP 向超级结点查找发送请求，然后超级结点执行查找过程，最后由超级结点在完成查找后，将查询结果返回给私网内的客户端。
- 与公网的客户端查找方式不同，私网内的客户端不与任何结点联系，说明客户端知道自己是处在限制 UDP 的防火墙后的。

4．呼叫建立与终止

在 Skype 系统中，用户之间呼叫的建立可分为 3 种不同的网络设置和两种不同的呼叫条件，其中，3 种网络设置为：

- 呼叫的双方都处在公网上。
- 呼叫的双方有一方在私有网络上，而另一方在公网上。
- 呼叫的双方处在不同的私有网络中。

两种呼叫条件为：

- 呼叫在好友列表中存在的用户。
- 呼叫在好友列表中不存在的用户。

需要注意一点的是，当被叫的用户不在主叫的好友列表中时，其呼叫过程其实包含了两个步骤，一个是查找，另一个是呼叫。查找用户的方法在上文已经讲过，所以关于呼叫的建立与终止，只讲一下用户都在主叫的好友列表中的呼叫建立的情况。

第 1 种情况：双方都在公网上、在线，并且都在对方的好友列表中的时候。在 Skype 中，当按下呼叫按钮后，呼叫方就与被叫方建立 TCP 连接，通话的信息通过 TCP 交换。最初主叫方和被叫方之间的消息交换表明了竞争响应机制的存在。主叫方同时也通过 UDP 发送消息来更换在注册过程中发现的在线的 Skype 结点。在这个过程中交换的数据量为 3KB。其通信过程如图 8.24 所示。

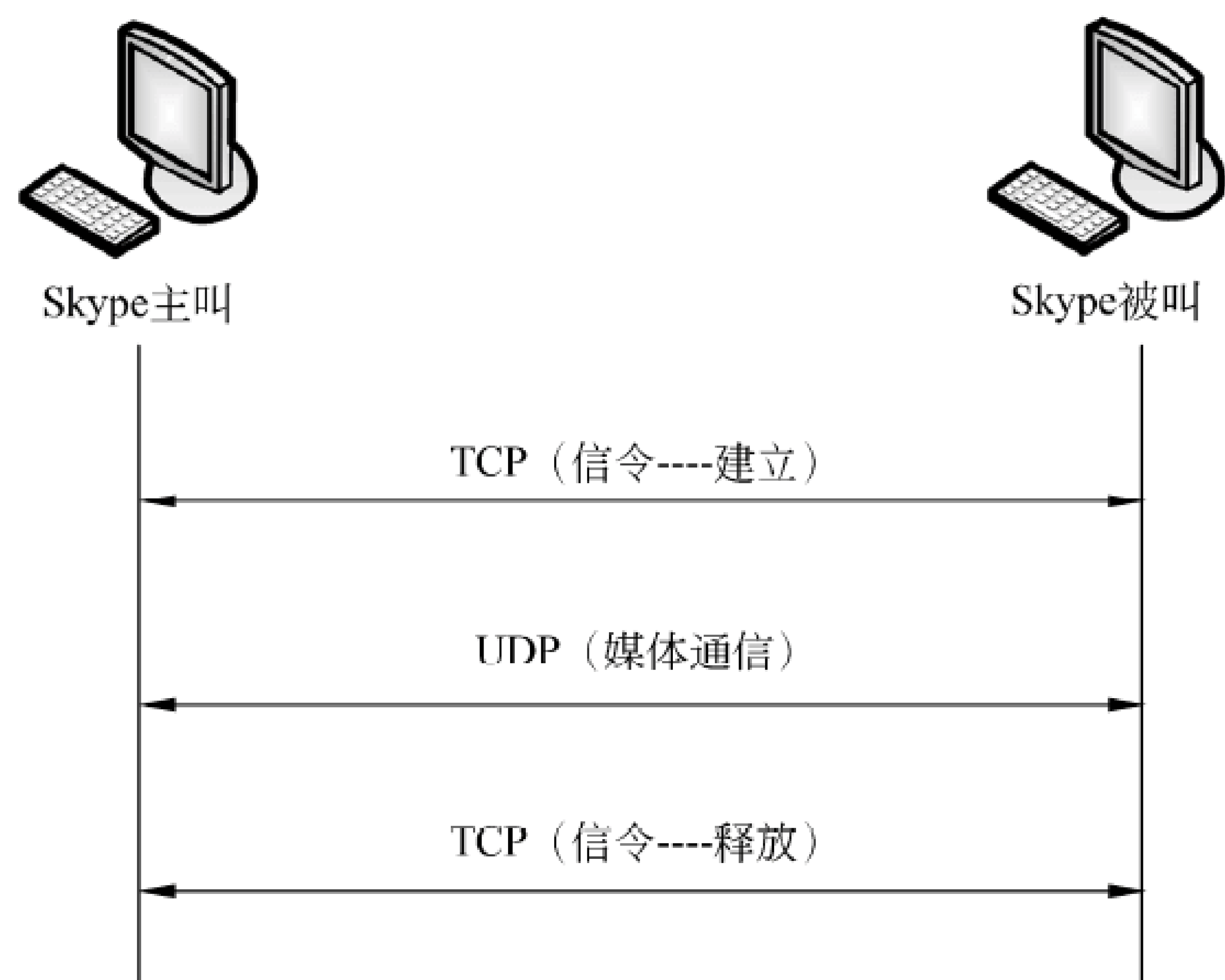


图 8.24　Skype 双方都在公网的时候呼叫建立与释放的过程

第 2 种情况：即主叫方处在限制端口的 NAT 后，而被叫方处在公网，这时通过 Skype 进行呼叫的时候，信号和媒体的传输并不是直接建立在主叫方和被叫方之间的，而是主叫方通过 TCP 发送信号给一个 Skype 结点，通过它来使用 TCP 转接给被叫方。这个在线的结点同时也通过 UDP 转发来自被叫者的语音数据包，反之也相同。主叫方发送在注册过程中 UDP 消息给结点并收到回复，说明主叫方就储存了这些结点的 IP 地址和端口。

第 3 种情况：即双方用户都处于限制端口的 NAT 和限制 UDP 的防火墙之后，主叫方和被叫方都通过另外结点的 TCP 连接转交控制信息。主叫方发送通过媒体信息给中间结点，然后由中间结点将信息通过 TCP 连接转交给被叫方，反之相同。

第 2 种情况和第 3 种情况有共同之处，就是双方之中有一方处在私有网络中，而且它们之间建立和取消连接的控制信息也是通过 TCP 传输的。在它们之间进行通信的过程中，都需要一个中间结点来进行消息的转接，其通信示意图如图 8.25 所示。

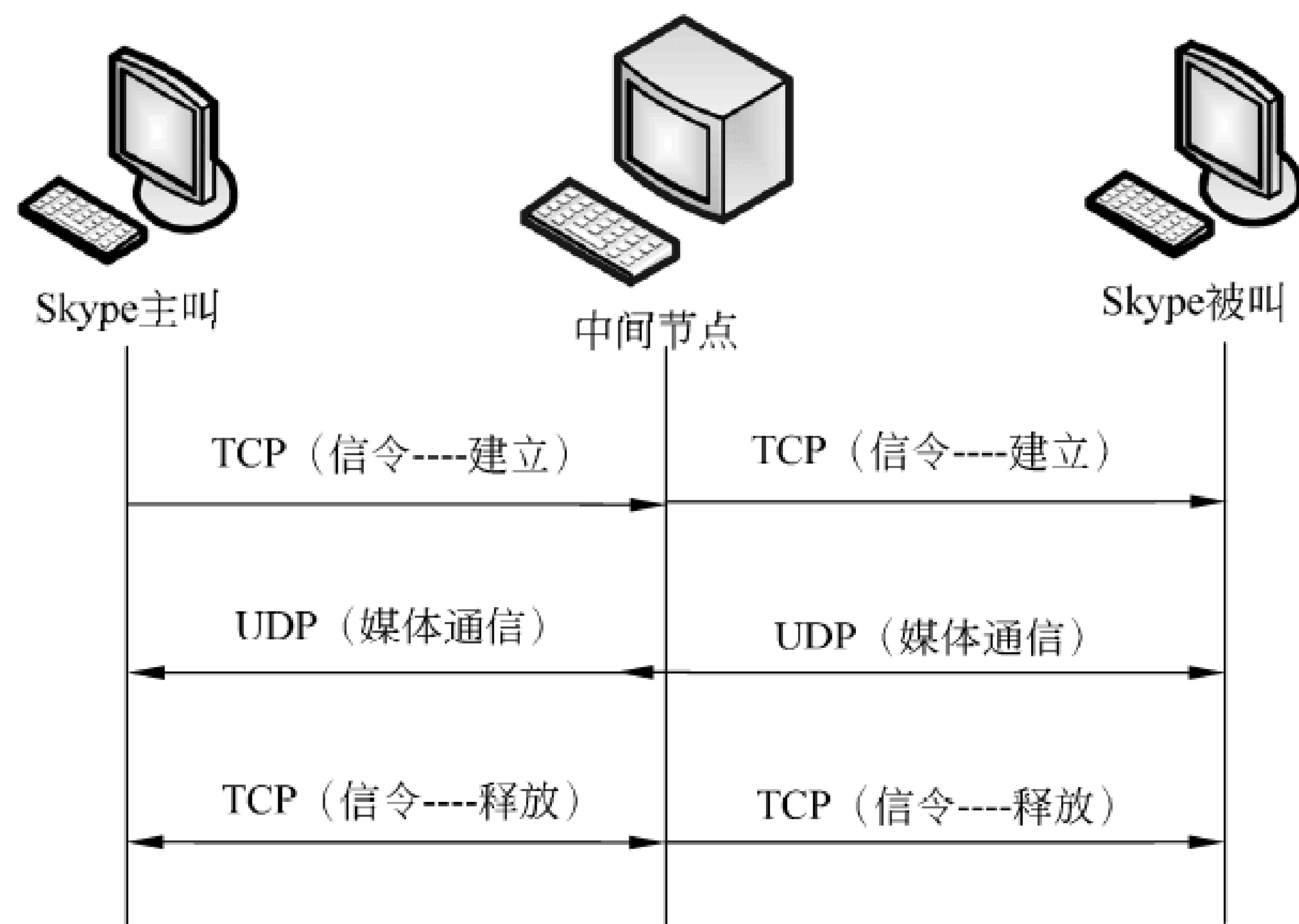


图 8.25　Skype 中至少有一方在私网内的呼叫建立与释放流程

如图 8.25 所示，使用中间结点为主叫方和被叫方中转语音数据包有很多优点。

- ❑ 它提供了一种处在 NAT 和防火墙后的用户建立通话的机制。

- 如果处在 NAT 和防火墙后的用户加入一个讨论组，同时公网的用户也想加入到其中，那么这个中间结点将作为一个服务结点提供混合和广播会议消息的服务。
- 不好的一点就是，将会有很多的信息流通过这个中间结点。当然用户一般都不希望任意的消息流通过自己的电脑。

总地来说，在 Skype 中用户之间呼叫的建立和终止是分不同的情况进行的。需要注意的是，Skype 的呼叫信令都采用 TCP 封装，而媒体流则使用 UDP 封装，当有任何一方的用户位于限制 UDP 包的防火墙内时，媒体流就会采用 TCP 封装。另外当 Skype 用户至少有一方位于私网内时，所有的信令和媒体消息都经过一个或多个中间结点转发。此时无须担心用户通话的数据流因为经过中间结点转发而被窃听，因为 Skype 采用了对消息进行端到端加密的机制。

5．媒体传输和编解码

在 Skype 中，媒体的传输过程也分为几种不同的状态，如果双方都位于公网中的时候 ，双方可以使用 UDP 数据包直接进行数据交换，Skype 的语音数据包的大小为 67 字节，即为 UDP 包的有效载荷。对于两个处在同一个 100M 以太网的用户，大约每秒可传送 140 个语音数据。一般来说，上下行语间传输所需的速率平均 5KB 每秒。这个带宽符合 Skype 宣布的 3～16KB 每秒的要求。

如果主叫方和被叫双方，其中有一方或者双方都处在私有网络中，就需要 TCP 同超级结点进行数据交换，Skype 客户端充当媒体代理服务器（中间结点）的角色。此时一个语音数据包的大小为 67 字节，正好是 UDP 的载荷。使用的带宽是 5KB 每秒。

在可能的情况下，Skype 会优先选择 UDP 协议进行通信。

6．保持连接控制消息

在不同的网络设置中，Skype 客户端每 60s 通过 TCP 向它的超级结点发送一次刷新信息，以保持同超级结点的连接控制信息。

Skype 是第一个利用 P2P 技术进行语音通信的 VoIP 工具，能够提供较好的通话质量。Skype 能够透过防火墙进行无缝通信，安装使用也很简单。随着互联网的不断普及，VoIP 技术已经取得了越来越多的应用，基于 P2P 的 Skype 技术也受到越来越多的重视，学习研究 Skype 技术有广阔的发展前景和巨大的潜力。

8.3.5　Skype 的安全机制

Skype 使用密码编码来验证用户和服务器的身份，并通过端对端的消息加密方式保护经过 P2P 网络传输的内容不被泄露。为此目的，Skype 设计了良好的安全机制和相应的密码系统来确保校验用户身份和实现在 P2P 网络中传输内容的保密。

1．Skype的安全政策

Skype 使用标准加密原语，核实每种加密技术，依照各自标准并相互参考逐一实施这些技术。

Skype 运行一个证书授权机构并授权，生产的数字签名是 Skype 用户身份的依据。Skype

使用一个专属的会话建立协议，此协议加密的目的是保护免受重放攻击，核实结点身份，并且使通信结点使用相同会话密钥，实现加密目的。总地来说，Skype 的安全政策如下。

- Skype 用户名是唯一的；
- 用户或应用程序在行使用户名身份或权限前必须提交 Skype 用户名和相关的证明凭证；
- 为建立 Skype 会议，各个用户都应正确地提供给其他用户用户名和权限证明。会议前每项被核实，才允许传输信息；
- Skype 会议传输的信息从 Skype 端到 Skype 端是加密的，没有中间结点。

2．Skype加密机制

Skype 的加密主要在体现在两个过程中，一个是注册（登录）的加密认证过程，另一个是端对端之间的消息加密传输过程。下面分别说一下这两个过程的加密机制，以进一步理解 Skype 的安全性能。

（1）注册/登录过程的加密机制

Skype 注册/登录服务器及验证过程如下。

- 用户首先写下自己选择的用户名和密码，然后客户端平台会产生 RSA 密钥对，即私钥和公钥（注意和服务器的 RSA 密钥对区别开）。用户选择的密码和用户名一起被 MD5 算法进行哈希得到。
- 接下来用户需要和服务器建立连接，客户端使用本地平台的一个特殊的随机数生成器（提供一个随机种子）生成一个对称的会话密钥。使用随机数的目的是为了在认证过程中避免遭到重放攻击。这个密钥被使用 SHA-1 算法进行哈希求值后得到一个 256 位的 AES 密钥。

客户端使用这个 AES 密钥 K，对上面得到的哈希之后的及用户的 RSA 公钥 Va 进行加密得到密文 Cipher；另外一方面，客户端还使用中心服务器的公钥 Vs 对会话密钥 S 进行加密，得到密文 Secret。两部分的密文 Cipher 和 Secret 被合并到一起传送给中心服务器。如图 8.26 所示为 Skype 中用户注册/登录服务器过程。

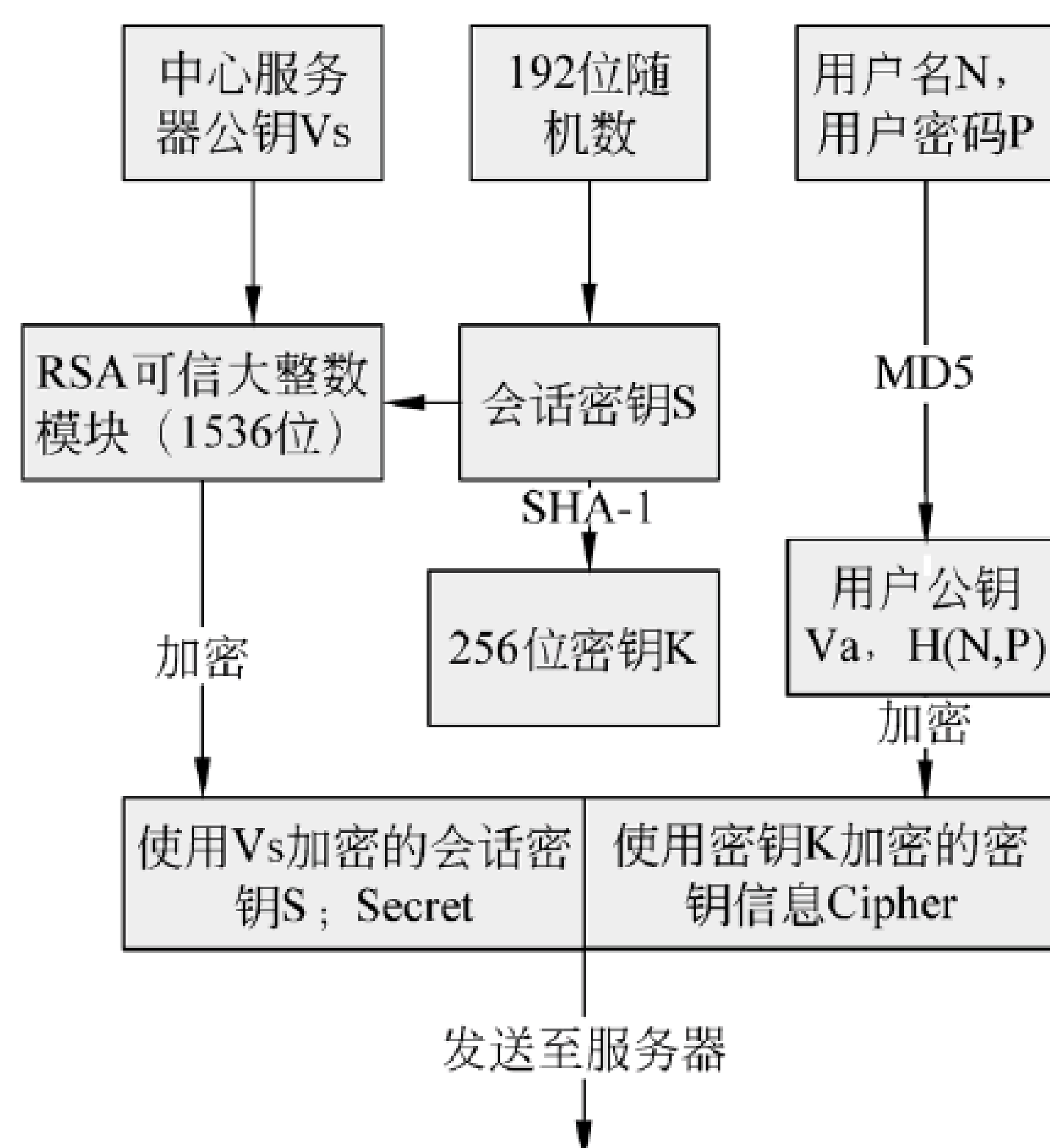


图 8.26　Skype 中用户注册/登录服务器过程

- 中心服务器首先使用自己的私钥 Ss 对 Secret 进行解密得到会话密钥 S，然后使用上文同样的 SHA-1 算法对其进行处理，得到 256 位的密钥 K。最后使用 K 就可以对密文 Cipher 进行解密，得到用户名和用户密码。
- 服务器首先检查用户名是否唯一，如果不是会要求重新命名；如果是唯一的就由服务器存储在数据库中。
- 服务器对用户名 N 及其公钥 Va 产生证书 ICa（IdentityCertificate），并且使用自己的 RSA 私钥 Ss 进行签名，返回给用户。其中包括用户名 N 和密码 Va 的绑定签名，Ss 的密钥标识符，以及用户的公钥 Va。如图 8.27 所示为中心服务器的验证过程。

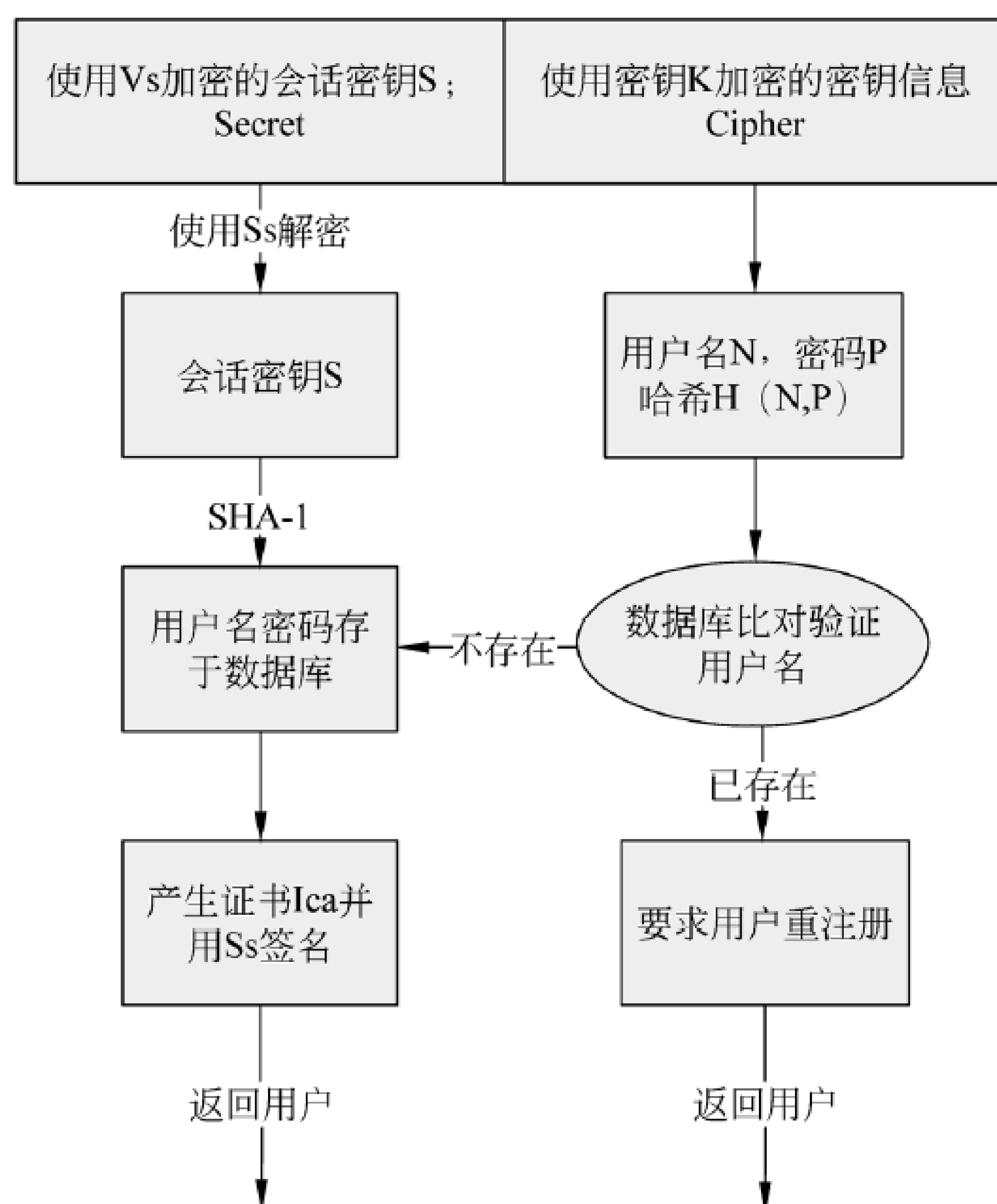


图 8.27　Skype 中心服务器的验证过程

（2）点对点通信过程的加密机制

Skype 客户端之间的点到点通信全部使用 256 位的 AES 算法加密，而在两个用户进行通信之前需要经历一个身份互相认证以及密钥协商的过程。

双方各向对方发起一个随机产生的 64 位数据的 challenge，这 64 位的数据也是由本地平台的随机数发生器产生的。另外，Skype 使用 RC4 算法。使用 RC4 算法加密的主要目的是对数据包进行混淆，使得一般的抓包工具（如 tcpdump、sniffer 等）无法分析出数据包使用的协议类型。

注意：RC4 算法，是一种加密算法，是由大名鼎鼎的 RSA 3 人组中的头号人物 Ron Rivest 在 1987 年设计的密钥长度可变的流加密算法簇。之所以称其为簇，是由于其核心部分的 S-box 长度可为任意，但一般为 256 字节。该算法的速度可以达到 DES 加密的 10 倍左右。

3．Skype的应用安全

从应用方面来讲，Skype 拥有比传统电话网络和 ISDN 语音交换高得多的安全性能，这些安全机制主要表现在以下几个方面。

（1）Skype 协议使用密文进行传输，因此即使监听者截获了数据，也无法轻易破解，无法得知通话内容和传输的文件。

（2）使用 Skype 服务的时候，每个用户都有全球唯一的用户名和密码，除非你的用户名和密码不幸被别人知道了，否则就可以保证通话的真实性。

（3）在使用 Skype 的时候，Skype 用户需要通过通信网络登录认证服务器验证用户名和密码。如果网络不可用，则验证无法进行，会话也就无法进行，因此网络的可用性是 Skype 服务可能性的直接因素，就网络的可用与否而言，无法对 Skype 的通信安全造成威胁。

（4）通信网络的存活性直接影响 Skype 能否继续使用。如果通信网络只有一条通路，那么该通路被破坏了，Skype 通信就无法进行。如果通信网络是两条冗余，那么一条被破坏了，Skype 会话仍然可以通过另一条完成。

（5）在伸缩性方面，当网络遭到意外破坏，通信暂时中断时，Skype 基于包交换网络比起传统的电话交换网络，可以在很短的时间内完成恢复。而且 Skype 通信用户有一个移动的概念，它不受物理地点的限制。当一个 IP 被占用或被攻击时，Skype 用户可以换成另外一个 IP;当一所大楼失火,Skype 用户可以带着 Notebook 搬到另一所大楼继续使用 Skype 通信，完全不受物理地址的限制。

（6）完整性传输的角度来看，Skype 在语音、即时传输和文件传输方面却表现的非常好，而且语音效果清晰，通话质量高。对于完整性系统而言，使用 Skype 整体上来说比较安全。

总体来说，不管是从 Skype 本身的安全机制，还是从 Skype 的应用来讲，Skype 作为一款基于 P2P 技术的即时通信软件，总体是安全的。

8.4　Skype 的应用及发展前景

目前，Skype 在全球拥有 5400 万用户，每日新增用户 15 万左右。在 2005 年全球最具影响力品牌排行榜上，Skype 软件品牌首次被评为第 3 名，排在 Google 之后。据权威的调查机构显示，商务人士对 Skype 的接受程度达到 70%以上。而美国 VoIP 通话中，35.8%的用户使用的是 Skype 。在国内，Skype 与 Tom.com 合作进行业务推广，目前 Skype 在中国已经拥有 340 万用户。另外据《时代杂志》报道，Skype 中国用户每天增加 45000 多名。

注意：以上这些都是官方公布的 2005 年的统计数据。

这些数字足以说明 Skype 的应用范围之广，影响程度之深。下面就具体讲一下如何在实际生活、工作中使用 Skype。

8.4.1　Skype 的下载、安装及注册

本文主要是以中文版的 Tom－Skype 为例来讲解 Skype 的使用方法。Tom-Skype 的安

装软件可到 Skype 的中文官方网站上下载 http://Skype.tom.com/，用户根据需要可选择下载不同的版本。下面以 Skype 简体中文版为例，说明一下从下载到注册的整个过程。

1．下载Skype简体中文版

到 Skype 中文版的官方网站 http://Skype.tom.com/download（Tom-Skype 的官方网站）下载最新的 Skype 简体中文版。Skype 软件是免费下载的，在下载页面可以直接单击下载 Skype，也可以下载 SkypeClient.exe 来引导安装 Skype。如果使用时出现了任何问题，请先注意自己是否使用了最新的 Skype 简体中文版。Tom-Skype 的官网中最新版的 Skype 界面截图如图 8.28 所示。



图 8.28　Skype 安装包的下载界面图

2．安装Skype简体中文版

下载完 Skype 的安装软件包后，双击刚才下载的.exe 安装程序，就可以进行软件的安装了。

开始安装之后，软件会提示选择所使用的语言，默认的是“简体中文”。如果需要安装为其他的语言，单击语言选择的下拉列表框就可以自行选择安装语言，如图 8.29 所示。

在正式安装之前请先接受许可协议，如果不同意许可协议是无法继续安装的。单击“下一步”按钮，就可以进行安装了，安装过程如图 8.30 所示。整个安装过程不要发生中断，单击“下一步”按钮一直到最后安装完成，如图 8.31 所示。

在图 8.31 中，直接单击“开始使用 Skype”按钮，即可完成全部安装。



图 8.29　Skype 安装时的语言选择



图 8.30　Skype 系统的安装过程界面



图 8.31　Skype 安装完成后的界面截图

3．注册为Skype用户

Skype 安装完成后，在正式使用 Skype 以前，如果没有 Skype 账户，则需要进行注册，可以通过客户端注册为 Skype 用户。图 8.31 中，单击“开始使用 Skype”按钮后，即自动进入用户注册界面，如图 8.32 所示。

在注册界面中输入用昵称、用户名、密码、重复密码等，接受“Skype 最终用户许可协议”后直接单击“下一步”按钮，如图 8.33 所示。

在图 8.33 中，输入个人资料信息，然后单击“下一步”按钮就开始进行注册。如果提示此用户名已经存在，则需要更换用户名后重新进行注册。

注意： 注册 Skype 账户的规则是用户名至少 4 个字符，以英文字母开头，不能包括空格；密码至少 4 个字符；建议一定要输入正确的邮箱地址，主要用来找回密码。

注册完成以后就可以在自己的电脑上使用 Skype 了。Skype 与其他即时通信系统一样，需要登录、需要查找好友、进行一些简单的设置。至于 Skype 如何登录、如何进行通话和

聊天，下文会有简单的讲解。读者在打开 Skype 界面后，也可以找到相应的在线帮助和支持文档自行学习 Skype 的使用方法。

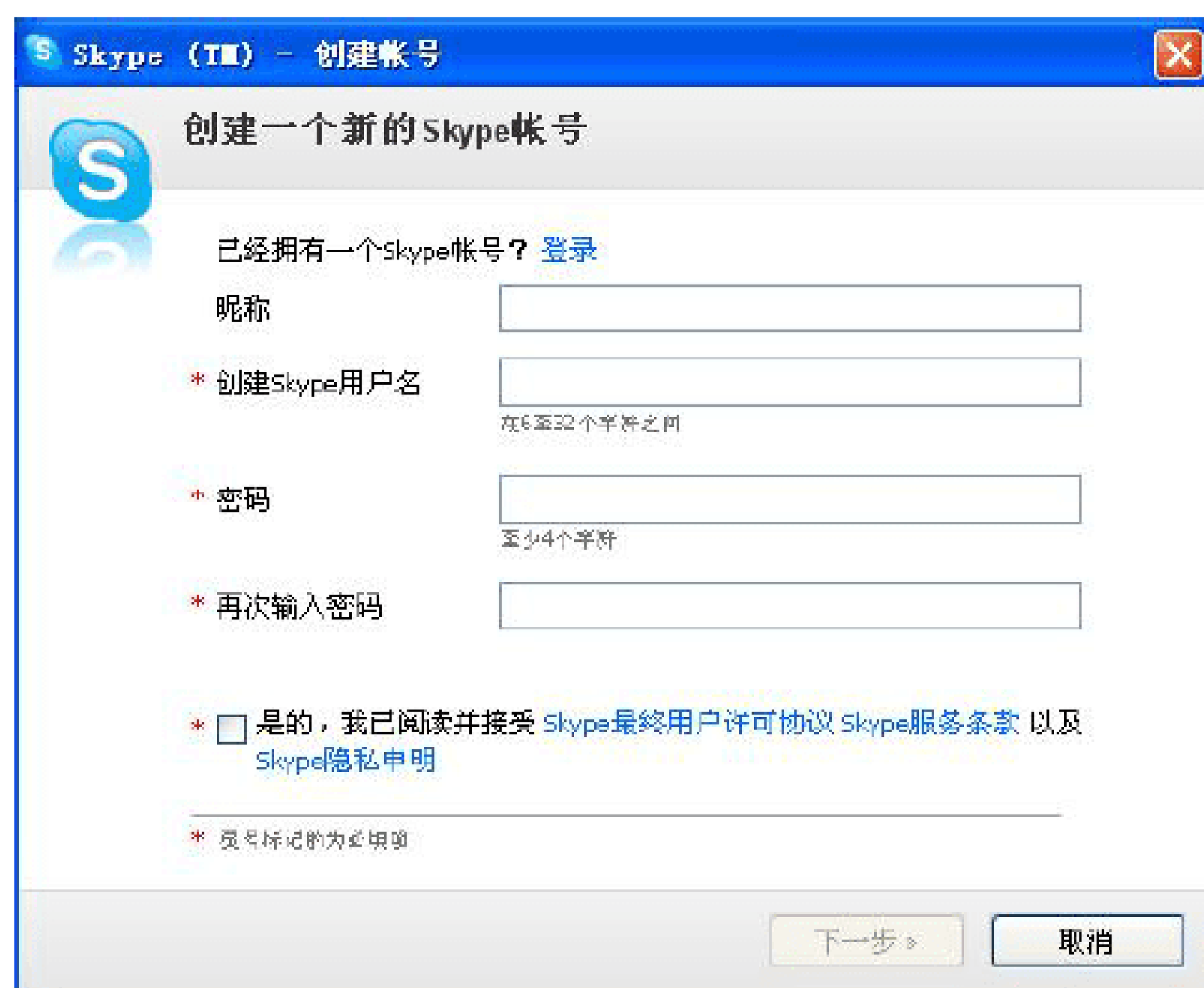


图 8.32　Skype 中用户注册界面



图 8.33　Skype 注册中输入个人资料信息界面图

注意：Skype 全中文的环境支持、友好的界面和简单的操作，可以使用户很快就进入 Skype 的世界。如果还有不明白的地方，请自行查阅 Skype 的在线帮助文档。

8.4.2　Skype 基本使用方法

Skype 无疑是最受欢迎的网络电话软件。Skype 电话不光占领 PC 市场，还有手机版 Skype 和 Skype 专用电话机，让你尽享 Skype 的清晰音质。Tom Skype 是 Skype 在国内的根据地。上文中已经介绍了 Skype 的安装和注册，下面就简单地说明一下，如何使用 Skype。

1. 使用Skype发送即时消息

打开 Skype 后，就会显示你的 Skype 好友列表，（没有好友的话，可以在 Skype 网络中搜索、查找），要想发送即时消息，在好友列表中选中一个你想发送消息的好友，直接鼠标右键，选择“发送即时消息”或者单击 Skype 界面上方的“会话”按钮，如图 8.34 所示，即可进入发送即时消息的窗口。

打开消息发送窗口以后，输入您要发送的文字或表情符号，然后按下 Enter 键或单击“发送”按钮，即时消息就会被发送出去了，如图 8.35 所示。

如果对方在线，那么他会立即收到您的即时消息；如果对方不在线，则消息会成为留言，在对方下次登录的时候自动发送给对方。

2. 查看即时消息历史记录

要在 Skype 中查看历史消息记录，方法也很简单，选择一位好友右击，然后单击右侧的“会话”标签，然后再单击“显示记录”，就可以查看所有的会话记录了，包括即时消息的历史记录，如图 8.36 所示。

单击“显示记录”的按钮以后，即可看到你与这位好友的详细聊天记录了，如图 8.37

所示的就是详细记录列表。



图 8.34　选择好友发送即时消息界面



图 8.35　Skype 中发送即时消息的界面



图 8.36　查看会话历史消息记录



图 8.37　Skype 中查看用户历史即时消息

注意：建议定期备份即时消息的历史记录，尤其是商务应用中的时候。备份方法是把“C:\Documents and Settings\登录电脑的用户名\Application Data\Skype”下，以你的 Skype 用户名命名的文件夹复制到其他盘（如 D 盘）即可。

3．使用Skype呼叫好友

Skype 最常用的功能就是进行语音通话，语音通话的第一步就是呼叫好友，进行语音呼叫是一个很简单的操作。

（1）选择要通话的好友，然后单击软件界面下方的绿色电话标志或者右击好友，选择带有绿色电话标志的“通话”选项，如图 8.38 所示。

（2）这时，软件主窗口将会切换到用户呼叫界面。如果对方在线的话，一会就会通了，则系统开始计时，当想结束通话的时候，只要点右下角红色电话标志就可以结束此次通话了，如图 8.39 所示。





图 8.38　Skype 中呼叫好友的操作方法　　　　图 8.39　Skype 中与好友进行通话的界面

4．用Skype拨打普通电话

Skype 除了用户之间通过网络进行语音通话以外，还有一个非常重要功能，就是可以拨打普通电话，在 Skype 中拨打普通电话是一项收费业务，所以在拨打普通电话前需要保证您的账户中有余额。

账户金额是通过购买 Skype 电话卡得到的，所以第一步就是需要买到 Skype 电话卡。

注意： 现在，越来越多的人都知道可以用 Skype 打电话了，Skype 是全球最清晰的网络电话，而且资费非常便宜。现在，用 Skype 拨打国际长途，最低仅为 0.187 元/分（按照 1 欧元=11 元人民币），国内长途最低为 0.11 元/分。比如往美国打 10 分钟国际长途，用普通 IP 电话，费用高达 24 元，用 Skype，花费不足 2 元。话费如此便宜，那么究竟如何使用呢？下面我们就一起来体验一下如何使用 Skype 拨打普通手机及座机。

（1）购买 Skype 电话卡

在拨打普通电话前，需要先购买一张 Skype 电话卡，并对你的账户进行充值，之后才可以通过 Skype 拨打普通座机及手机。先登录 Skype 中文官方网站 http://skype.tom.com ，进入“购买电话卡”页面；也可以通过 Skype 客户端界面上方“Skype-->购买 Skype 电话卡”，如图 8.40 所示，进入购买页面。然后选择自己想要的种类（国际卡或国内卡），单击“我要购买”，根据页面上的提示即可完成购买流程。

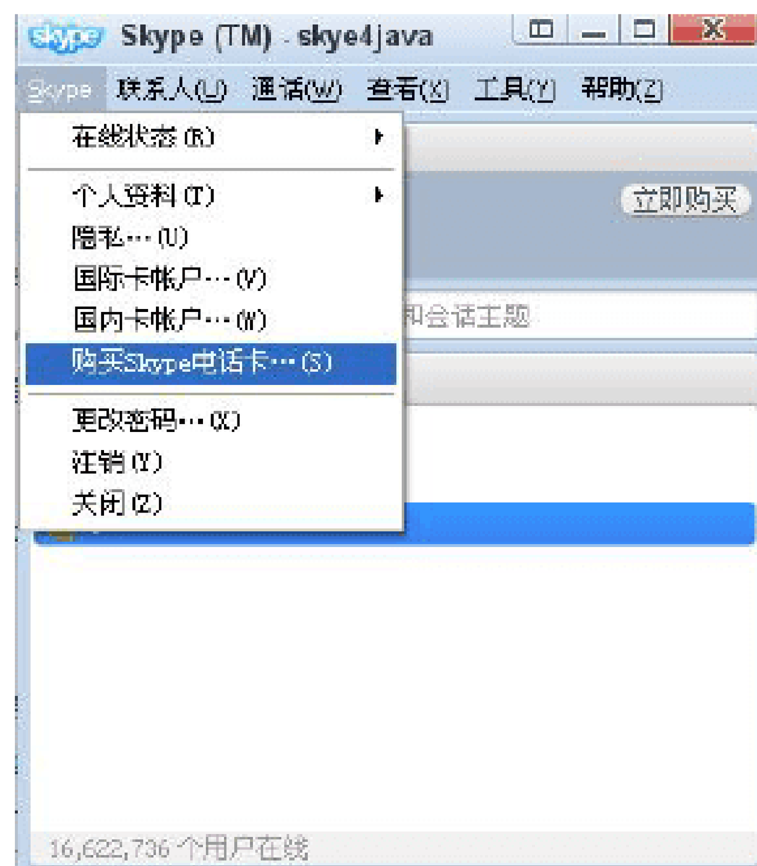


图 8.40　Skype 中购买电话卡的操作界面

注意： 需要购买 Skype 电话卡的用户，网上有详细的流程和购买步骤，这里就不再多说了。

购买了电话卡并完成充值后，会在主界面上方看到你所拥有的国际卡余额或国内卡时长，使用 Skype 打电话的时候，Skype 系统会有十分详尽的帮助信息，帮助你顺利地完成 Skype 通话的功能。

（2）拨打普通电话

购买了 Skype 电话卡，且有一定余额的情况下，就可以拨打普通电话了。如果只购买了 Skype 国际卡（无国内时长），请按如下方法拨打电话：

方法一：登录 Skype，单击“拨打电话”标签，选择国家名称，单击国家的国旗标签就会出现世界各个国家的列表。在接下来的空格中输入手机或座机的号码即可（拨打座机需加区号），如图 8.41 所示。

方法二：如果记得目的国的国家代码，可以按“00+国家代码+区号+对方的固定号码”格式输入对方的电话号码。比如拨号 0086108888888，86 是中国的国家代码，10 是北京的区号（而不是 010），88888888 是电话号码。拨打手机号码输入 00+国家代码+对方的手机号码，比如拨号 008613900000000，86 是中国的国家代码，13900000000 是手机号码。呼叫号码格式示意图如图 8.42 所示。

如果你同时拥有 Skype 国内卡和国际卡，那么在拨打国际长途时可按照上述方法一、方法二。但在拨打国内长途时，则务必在号码前加*，以确保国内长途是从国内卡中扣费。

注意： 拨打国内电话时，务必使用最新的简体中文版 Skype 软件，以确保你在全球任何地区都能正常使用 Skype 国内卡（可到 Skype 中文官网下载最新版本 http://skype.tom.com）。



图 8.41　用 Skype 国际卡打普通电话的方法



图 8.42　Skype 中呼叫普通电话号的格式示意图

以上就是用 Skype 打电话的步骤，其实非常简单，不过关键之处在于它是一个节省电话费的好方法。

5．进行Skype多人语音会议

Skype 的多人语音会议，简单地说就是一个多人的电话会议，此功能可以很方便地进行多人同时通话。操作方法如下。

在 Skype 的界面中，选择“联系人”|“新建组会话”命令，如图 8.43 所示。

或者直接单击 Skype 界面的“新建”按钮，然后也是选择“新建组会话”选项，也可以完成同样的功能，如图 8.44 所示。新建一个组会话以后，就会弹出如图 8.45 所示的对话框。在图 8.45 所示的组界面中，可以直接将要进行组会话的好友拖曳进去，也可以单击图 8.45 中所示的“＋”图标，从用户列表里进行添加，如图 8.46 所示。



图 8.43　Skype 中新建一个组会话



图 8.44　Skype 中新建组会话的另一种方法

在图 8.46 所示的界面中，选择要参加会议的好友，单击“添加”按钮，就可以添加一个用户到组会话中了，也可以通过“删除”按钮，来删除组会话中的用户。所有用户添加完毕后，再单击“呼叫组”按钮，如图 8.47 所示，即可开始多方语音会议了。

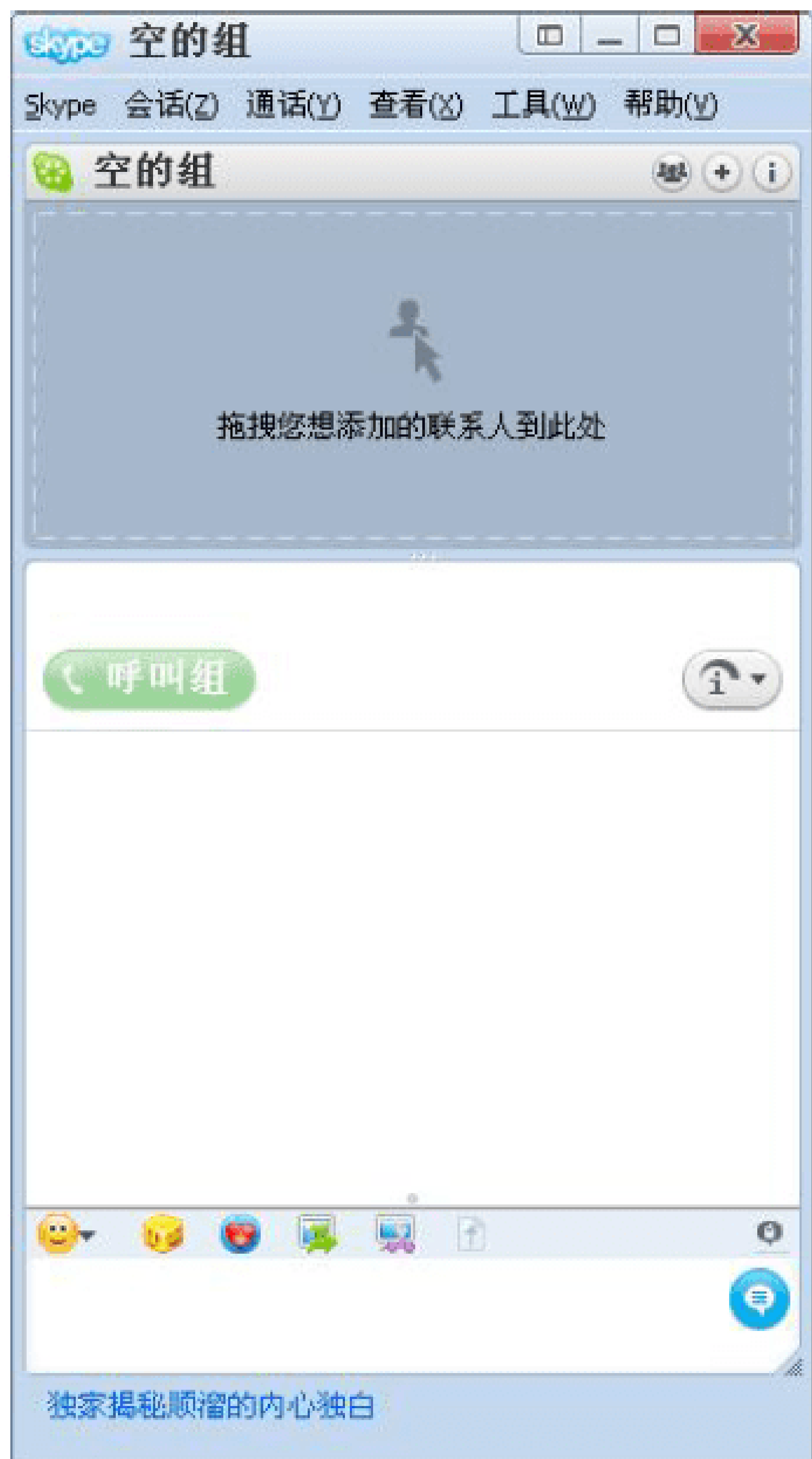


图 8.45　Skype 中的组会话

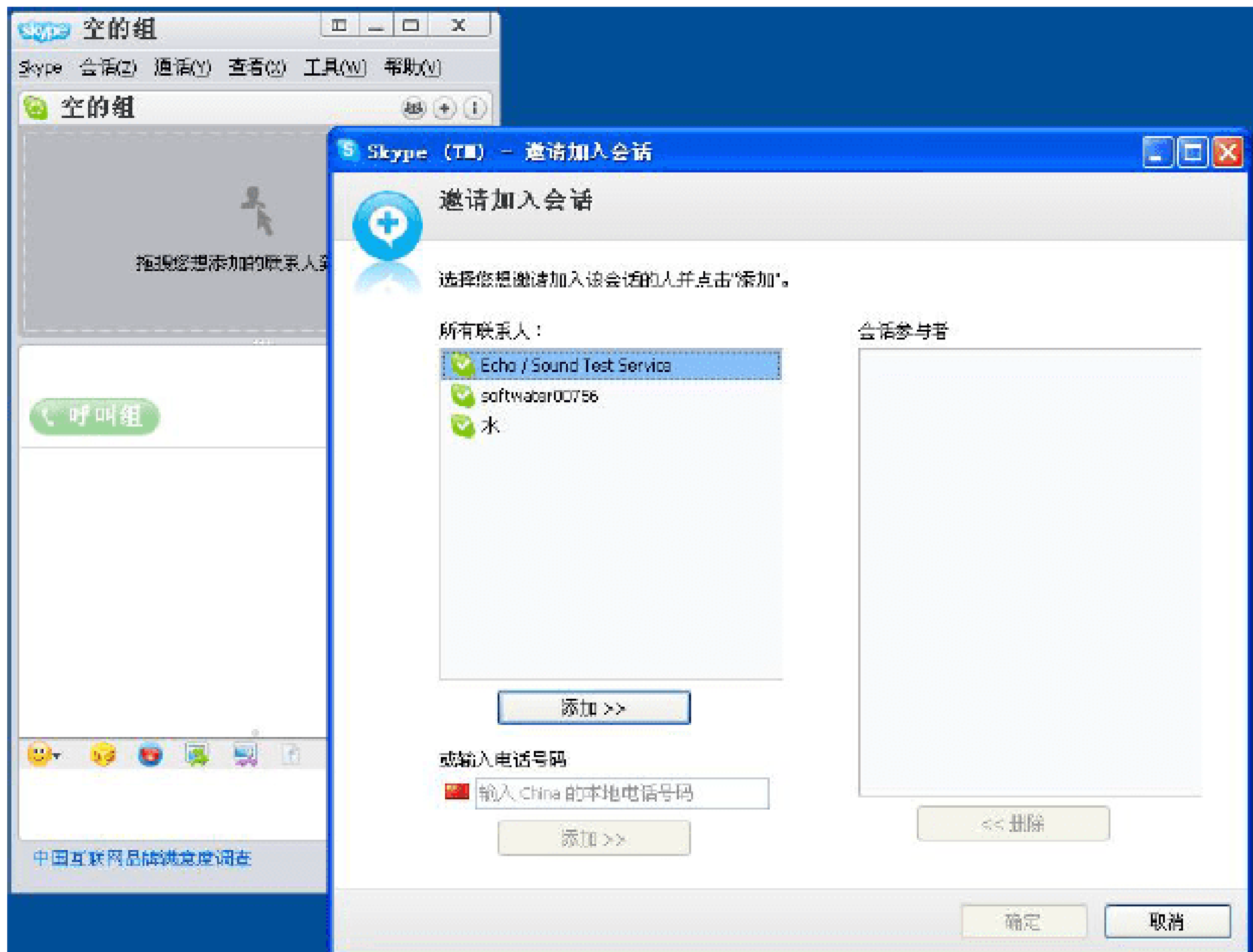


图 8.46　从好友列表中添加用户到组会话中



图 8.47　Skype 中多方语音会议的工作界面图

以上所介绍的就是 Skype 即时通信系统的基本使用方法。Skype 是一个非常强大的即时通信工具，在实际应用中，它的很多功能还需要在使用中不断摸索、不断钻研。Skype 在线帮助系统提供了各种操作方法的说明，有不明白的地方，查阅在线帮助文档即可。

8.4.3　Skype 的高级应用

目前绝大多数的 Skype 用户都只是 PC2PC 的个人用户，真正在企业通信系统中应用 Skype 的用户却很少，这种情况主要是由于没有适合企业使用的 Skype 应用设备。如何利用现有的设备，以及开发出适合企业使用的设备，是扩大 Skype 在企业通信中应用的关键。

1．构建Skype的企业级应用网络

通常，电话交换机或集团电话是企业中必备的通信设备，如果将 Skype 与这些通信设备结合，就可以实现企业中 Skype 的应用。如图 8.48 是一个企业中简单的 Skype 应用。

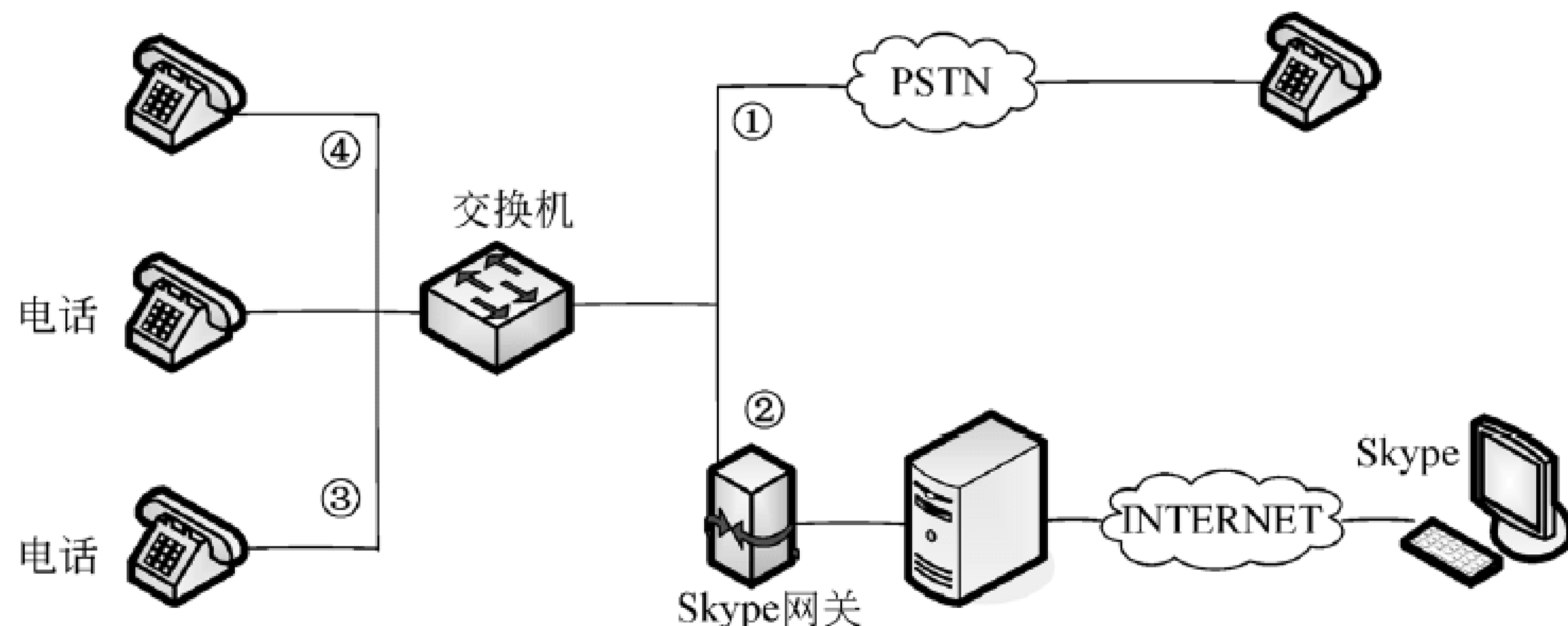


图 8.48　Skype 构建企业级应用的网络模型

利用一个 Skype 网关接入电话交换机的外线端口，用户就可以通过互连网打入交换机。用户通过互联网呼叫接在交换机上的网关（2）这时总机（3）开始振铃，接线员接起电话就可以与 Skype 用户通话，如果要与其他的分机通话，可以通过总机转到其他的分机，这样用户就可以与电话交换机的所有分机通话。

2．Skype常用设备的接入方法

（1）Skype 耳麦

Skype 耳麦是 Skype 用户利用计算机中的声卡，通过耳机和麦克风与互联网上的其他用户通话是最简单的 Skype 应用。不管路有多远，只要有网络，用户就可实现免费通话，如图 8.49 所示，是 Skype 耳麦的接入方式。

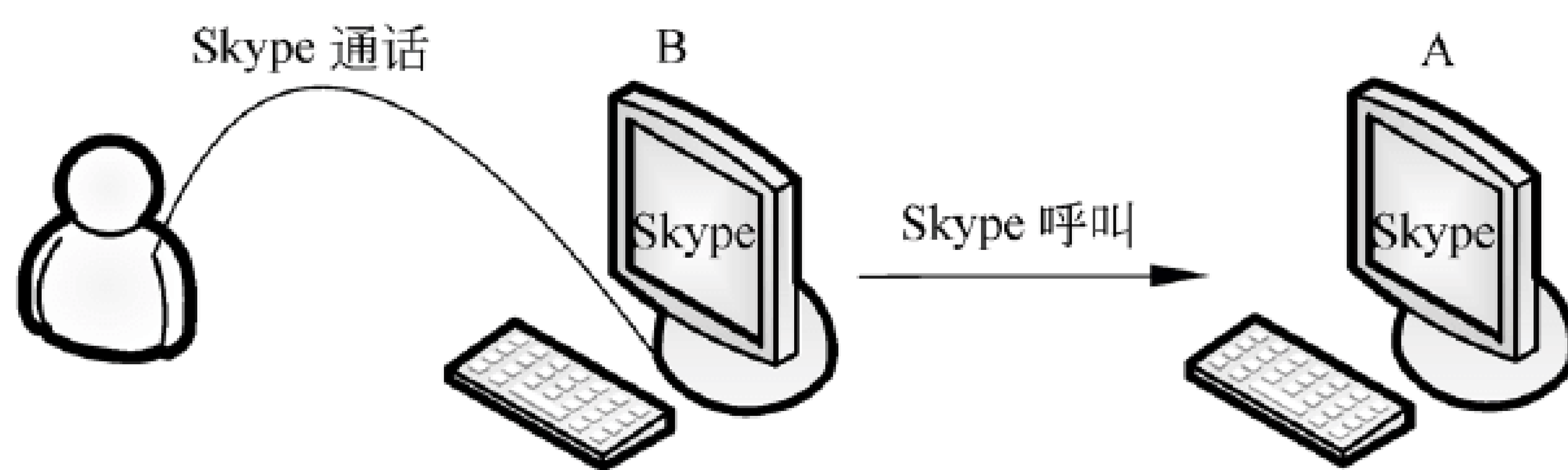


图 8.49　Skype 系统中 Skype 耳麦的接入图

（2）Skype 专用话机

如图 8.50 所示，Skype 用户使用 Skype 专用话机与互联网上的其他用户通话。表面上

看，这种通话方式与上一种通话方式不同，是通过话机与 Skype 用户通话，就像拨打普通的电话一样。但是，这种通话方式与上一种通话方式完全相同，它是将计算机中的声卡、耳机和麦克风集成在 Skype 专用话机中。通过计算机中的 USB 接口，外接 Skype 专用话机实现 Skype 用户间的通话。

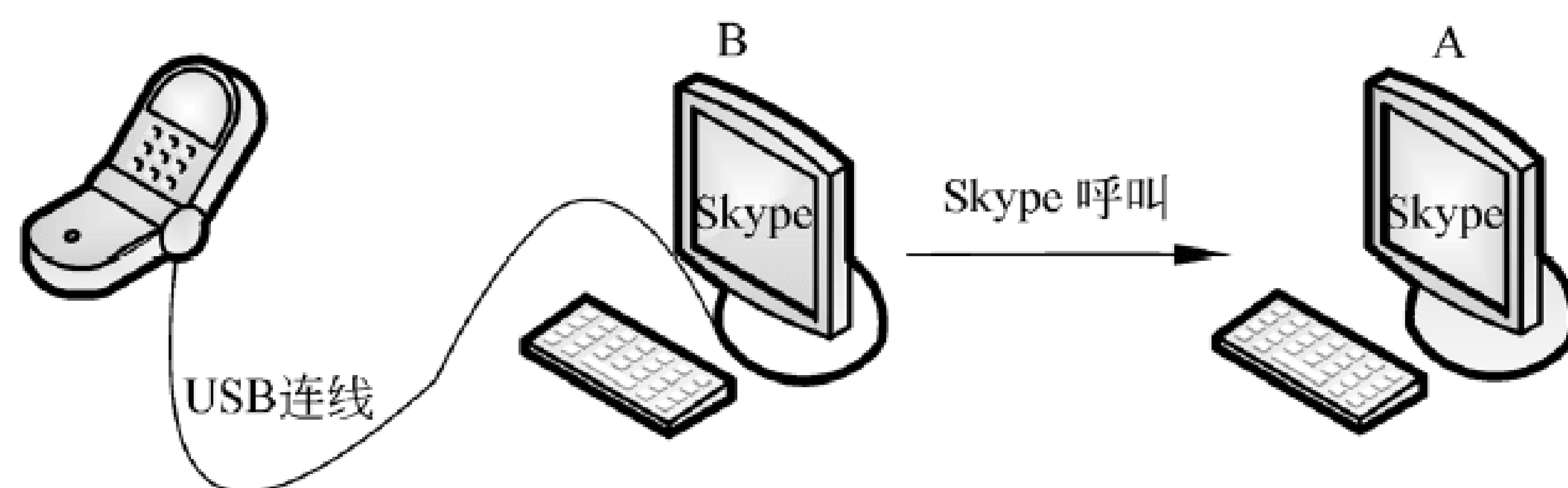


图 8.50　Skype 系统中 Skype 专用话机的接入图

（3）Skype 电话

Skype 电话：这种通话方式与上述两种通话方式有所不同，用来通话的话机是普通的电话机，而不是 Skype 专用话机，并且接线方式也有所改变。外接 USB Skype 转接盒通过 USB 连接线与计算机的 USB 接口相连。外接 USB Skype 转接盒除了有 USB 接口外还有两个 RJ11 接口，即一个接电话，另一个接电话线。接入示意图如图 8.51 所示。

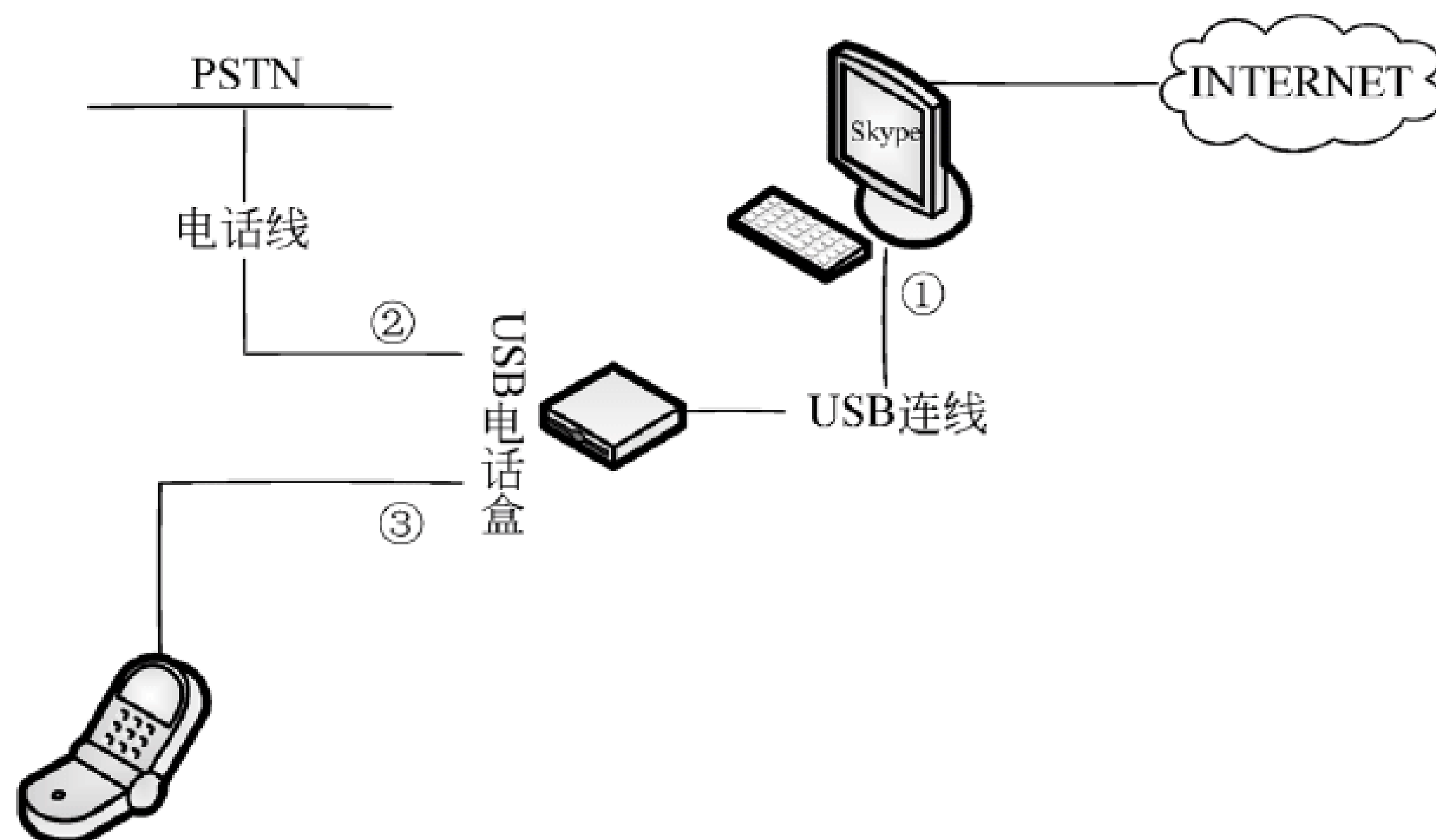


图 8.51　Skype 系统中 Skype 电话的接入图

在这个方案中，Skype 转接盒有两个用途。通过连接的普通电话机，可以照常打固网电话。由于 Skype 转接盒已接到电脑的 USB 接口，只要在 Skype 转接盒软件里，将使用模式由 PSTN 改为 USB，那这个电话便可以当作 Skype 专用话机一样使用，这样就可以省下耳机和麦克风了。

3．Skype网关

Skype 网关：有时候，用户未必经常在电脑旁边，那便会有可能接漏了电话。在图 8.51 中将 USB 转接盒换成 USB Skype 网关，用户就可以在其他地方接听来自 Skype 的电话。

在这个方案中，用户只要在 Skype 应用网关上先设定好一个号码，例如手机的电话号码。这时，如果有其他用户（A）拨打进来，Skype 应用网关（B）没有回应时，Skype 网关就会替你将用户（A）的呼叫转移到手机或电话（C）上，那样即使你在街上，仍然可以

通过手机（C）跟 Skype 用户（A）通话。通信过程如图 8.52 所示。

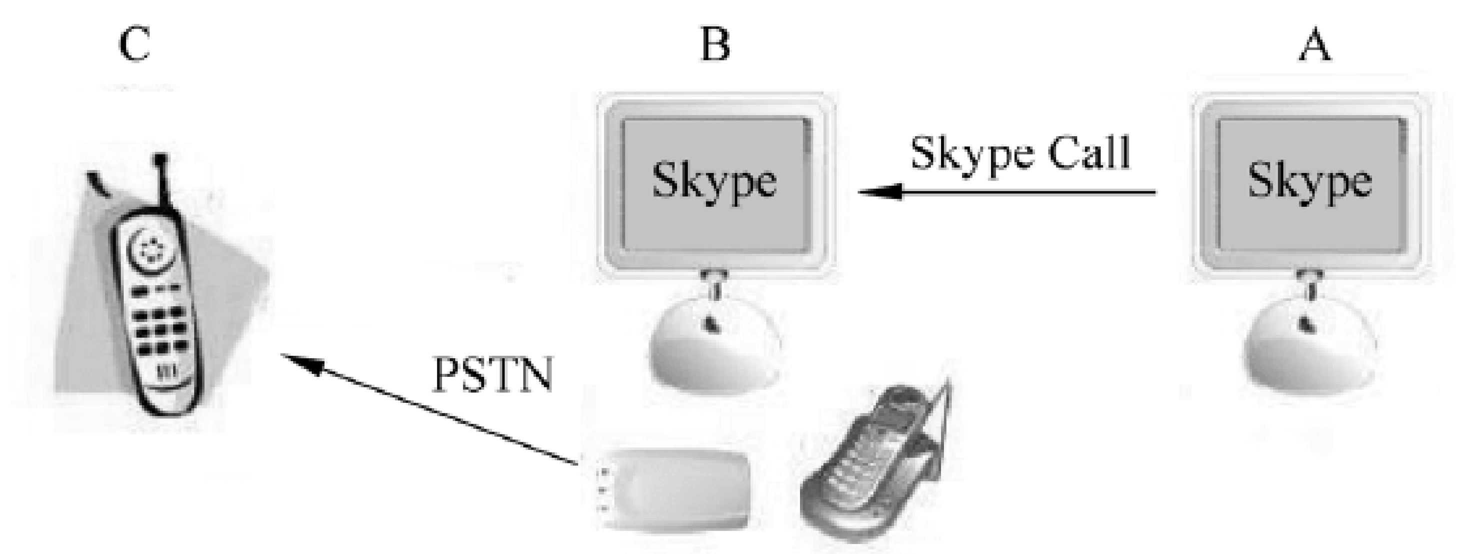


图 8.52　用户接听来自 Skype 网关的电话示意图

反过来，你也可以通过电话（C）打入网关（B），如果 Skype 应用网关没有应答，网关就将电话呼入转移到用户（A）上，这个用户是你在 Skype 应用网关上预先设定好的一个速拨号码。也可以按照网关的提示，输入密码，如果密码正确，就可以输入用户（A）的速拨号码，呼叫网上的用户。这个过程如图 8.53 所示。

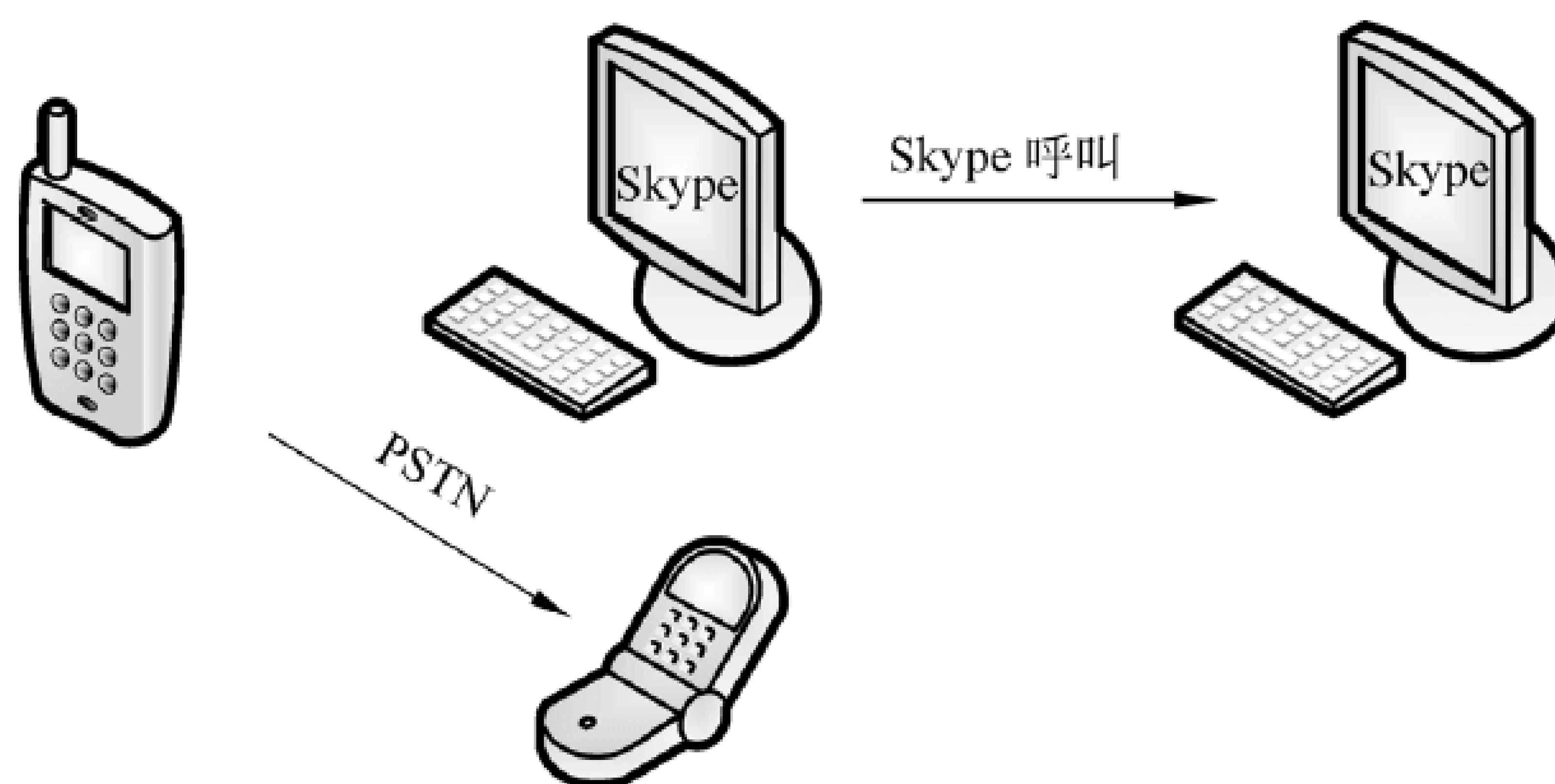


图 8.53　用户通过电话打入 Skype 网关示意图

实际上，这种通话方式还不是真正意义上的电话网关，只不过是通过呼叫转移的方式将呼叫转移出去，所以，一般称这个 Line 端口为逃生口。

注意：以上的这些设置需要专业的外接设备，以及开通相关的服务才行。个别功能针对不同的平台和特殊需要，还需要安装相关的 Skype 插件才能使用。有实际这方面的应用的读者，请参阅其他相关的说明。

8.4.4　Skype 在商业上的应用及发展

每天全世界近万用户在拨打网络电话，在没有任何广告的情况下，一个月间它已经被下载超过亿次，每天增加数万用户的增长速率还在提高，通话的时间长度已经高达亿分钟。这样的增长速度和业绩令人瞩目，震撼了世界其他的科技公司，微软、雅虎等巨人也群起仿效相继投入网络电话战场。

全球各大电信巨人更是紧张，一些传统运营商甚至筹措对之堵截封杀。与普通电话相比，拥有清晰的话音质和远远低于普通电话的通话费用，对那些寻求如何降低自己打电话

成本的用户而言带来一场真正的全球通信革命。

1．Skype的应用模式

Skype 的通信模式分为 Skype、SkypeIn、SkypeOut 共 3 种，它们分别对应 PC-PC、Phone-PC、PC-Phone 通信模式。对于第 1 种模式，因为多款 IM 软件也提供类似的功能，且通话质量也不理想，因此中国电信运营商及监管机构一直持默认态度任其发展。当然，这并不是 Skype 的特长。

对于 SkypeIn 模式，由于通话主叫方仍然用电话，对电信运营商的收入影响不大，因此也没引起中国电信运营商的抵制；

然而对于 SkypeOut 模式，中国电信运营商的态度与前两者相比却截然不同。因为通过 SkypeOut 模式打国外长途的花费只有传统电话打长途的几十分之一，显然 SkypeOut 将大大影响到国内某些电信运营商的收入。

从这几种模式中也可看出，用 Skype 的 SkypeOut 模式打电话，与传统的电话相比有着巨大的优势。

在“通话费”上的比较。Skype 电话，对于用户来说，它最大的诱惑力莫过于话费开支的大大降低。中国电信的传统电话打美国、英国、加拿大国际长途每分钟需 4.8 元，用 Skype 卡电话打这 3 个国家的电话，除市话费支出外每分钟约需 0.187 元，这种接近“零话费”的通话费用也是让 Skype 大受欢迎的一个重要原因。

在“通话安全”上的比较：在日常生活和工作中，无论是传统模拟电话还是现在被广泛应用的 IP 电话，我们常常会遇到“电话串线或通话被窃听”的尴尬。而 Skype 电话就可以有效解决通话中的一些顾虑，Skype 的网络特性及其安全机制，使得它在保密性、安全性、可靠性等方面独占鳌头。在通话安全上，远远优于普通电话。

2．Skype的领先优势及巨大商业价值

2005 年 8 月 29 日，Skype 公司迎来了它两周岁的生日。两年来，这家公司已经以擅长提供基于 VoIP 技术的电话服务和 PC to PC 间的网络电话而在业界享有盛名。

总部位于卢森堡的 Skype 公司由瑞典人 Niklas Zennstr 与丹麦人 Jaunts Friis 联合创立。截至 2005 年 8 月 4 日，全球有超过 180 万人在使用 Skype 一项名为 SkypeOut 的付费服务。这项服务可以帮助用户通过互联网与传统电话机通话，而且费用比较低。同时 Skype 公司的软件已经被近 1.45 亿用户下载，全世界有 4700 万人在使用它们的服务，到 2005 年 11 月份时这个数字已经上升到 6 00 万，并且还在以每天 17 万的速度增长。

Skype 取得的成功，与其技术领先、设计简单、价廉实用是分不开的。

（1）在技术上 Skype 并非完全自主研发，而是采用了一些成熟的技术，例如，分别采用了 Joltid 和 Global IP Sound 的 P2P。

（2）Skype 可运行在 Windows、苹果 MAC、Linux 和 PPC 便携设备上，而现在流行的 IM 类软件一般都只能在 Windows 平台上运行。Skype 同时还可以在 Mac OSX、Linux 和 Pocket PC 平台的 PDA 上使用，并且在不同的平台上有适合本平台的界面，不会因为使用不同的操作系统而无法与其他用户进行沟通。

（3）在产品设计上 Skype 使用简单。其用户群范围定位为全球用户，采用了多语言的形式，不仅能使得全球用户轻易上手，而且操作简单；并且 Skype 提供了方便的发送好友

列表、推荐给好友等便于共享的方式，可以让用户轻松上手。用 Skype 流传甚广的一句话说“连老太太都会用”——只要下载安装申请一个账号，添加一个好友，带上耳机就立刻可以开始语音通话了。

（4）Skype 最大的优势还是在价格上。用 Skype 打固定电话国际长途仅需要市话费用，并且语音效果极佳、接通率高，所以它不仅赢得了个人用户，还有部分商务用户。

仅仅推出 5 年，Skype 就成为全球最大的国际语音通信供应商。事实上，Skype 经常被视为电信运营商的对手，后者担心基于 IP 的网络通话将进一步减少其营收。

2008 年，Skype 提供的收费 Skype Out 服务（可以实现用户拨打普通电话）通话时长为 84 亿分钟，约占 Skype 所有国际通话量的 1/4。最近，Skype 更是与和记黄埔在英国的 3G 业务方面达成合作。由于很少用手机进行非商业国际通话，所以利用 Skype 打国际长途对移动运营商影响很小。

毫无疑问，迄今为止 Skype 主要面向个人用户，不过增长趋于稳定的个人用户市场已经不足以支持 Skype 的发展，因此 Skype 开始把目光转向商业市场。2009 年 4 月初，Skype 公布了一项互联网电话软件新版本，可以与企业 IP 电话系统连接。这项名为 Skype for SIP 的新服务有望使员工通过普通办公室电话以廉价的费用拨打国内、国际电话而无须将头戴式耳机接入 PC。

Skype 于 2005 年 9 月 12 日被网络拍卖巨擘 eBay 公司以 26 亿美元（约新台币 849 亿元或合人民币 210 亿元）的现金跟股票并购。初期将支付 13 亿美元现金与价值 13 亿美元的 3240 万股 eBay 股票，若 Skype 在 2008 年或 2009 年达到业绩目标，eBay 可能要再支付 15 亿美元，使得并购总金额高达 41 亿美元。

在 2008 年 4 月，公司推出了用于 Windows Mobile 的 Skype 软件。此软件基于 Java 开发而成，可以接收 Skype 和 SkypeIn 的呼叫，也可以收发即时消息。

8.4.5　中国的 Tom-Skype 简介

Skype 是一家互联网通信公司，母公司是 eBay；Skype 与 Tom 在线在中国设立了合营公司 Tom-Skype，而 Tom 在线是这家合营公司的大股东。

TOM-Skype，是 TOM 集团和 Skype 于 2004 年 11 月在中国大陆联合推出的 Skype 中国大陆版即时通信软件。截至 2007 年 6 月底，TOM-Skype 注册用户已经突破 4200 万，超越美国成为 Skype 全球最大的市场。

注意： 在中国大陆，Skype 与 TOM 集团旗下北京讯能网络有限公司 TOM 在线合作，所推出的 Skype 又称为 TOM & Skype。在台湾是与网络家庭（PChome Online）合作，推出的 Skype 称为 PChome & Skype。在香港，Skype 与和记环球电讯合作，推出的 Skype 称为 HGC-Skype。在日本则与 livedoor（活力门）合作（与活力门的合作已于 2008 年 1 月 8 日结束，现与 Buffalo 和 Excite 合作）。

网络电话（Volp）软件巨擎 Skype 在中国的注册用户数已突破 2500 万，高居全球市场第一。Skype 认为，明年其注册用户数将凌驾微软（Microsoft）的 MSN。

Skype 中文官方网站负责人表示，两年前 Skype 与 TOM Online 携手合作推出中文版软件 TOM－Skype，而今已成为中国市场成长最为快速的即时通信（IM）软件，每季度增

长率达 1.00%。

目前 Skype 全球注册用户数逾 1.3 亿，在中国市场，TOM－Skype 的用户数超过 2500 万，占 Skype 全球用户总数比例逾 20%。

目前即时通信市场竞争激烈，中国本土的腾讯 QQ 与微软 MSN 维持领先地位。而 TOM－Skype 将继续强化功能，以简单、实用的需求，在 IM 市场上获得更大的市场占有率。

Skype 的技术及其应用，不管是在中国还是在世界，都有着巨大的价值，掌握了 Skype 的核心技术也就掌握了资本、掌握了财富。

8.5　本章小结

作为一种新兴的网络技术，Skype 融合了 VoIP 的“话语权”和 P2P 的技术优势，从其诞生那一天开始就充当着一种颠覆者的价值，不论是从技术、应用还是从商业的角度，Skype 给即时通信领域甚至整个互联网都带来了焕然一新的感觉。

本章从即时通信的基本知识讲起，带领读者了解了整个即时通信的知识体系。接着讲解了 Skype 的相关知识，更详细阐述了 Skype 在 VoIP 和基于 P2P 的网络结构方面所展现出的卓越表现。正是 Skype 的出现，将过去通过互联网的通话技术推上了一个新台阶。然后详细分析了 Skype 的协议内容和完整的通信流程，从更底层的角度学习 Skype 的技术实质。最后介绍了 Skype 的基本应用，告诉读者如何使用、用好 Skype 系统。

Skype 不管是在过去、现在还是将来，都有着极强的生命力和巨大的发展前途，学习 Skype 技术有着重要的价值和意义。

第 9 章　基于 P2P 的流媒体技术

流媒体业务正变得日益流行，然而传统流媒体系统存在着种种不足。新兴的 P2P 技术能够利用客户端结点的资源减轻服务器和骨干网的负担，能够利用结点之间交互的特性为解决流媒体内容分发提供了一个新的方向。当前，P2P 流媒体技术已成为网络应用中热门的技术之一，具有重要的研究和应用价值，本章将重点讲解一下基于 P2P 的流媒体技术。

本章主要讲解的知识点如下。

- 流媒体的基本知识：了解什么是流媒体，理解流媒体的技术原理和基本的传输方法，简要了解流媒体的主要应用及所面临的主要问题。
- 流媒体与 P2P 技术的结合：重点掌握 P2P 技术应用到流媒体上的特性及 P2P 在分发流媒体数据上的基本特点，同时要了解 P2P 流媒体技术广阔的应用前景。
- P2P 流媒体的技术分析：理解 P2P 流媒体的几个重要的、关键的实现技术，如分发策略、编码技术、定位技术、媒体同步、数据交换及缓存等，还要掌握流媒体系统的几个重要的机制。
- 本章最后要了解一下 P2P 流媒体技术的主要应用，以及典型的应用平台。

9.1　初识流媒体

在 Web 网络的早期时代，互联网上只能传送一些静态的文本信息和图片信息，当人们在享受网络的好处时也不无遗憾，因为在当时的条件下还无法传输有声有色的多媒体信息，特别是电影、电视等视频信息。直到宽带流媒体技术的出现，才实现了人们在“在互联网上看电视、看高清大片”的梦想。那么什么是流媒体技术呢？本节就将带领读者学习流媒体技术的基本知识。

9.1.1　什么是流媒体技术

在互联网发展到一定阶段的时候，利用网络传输声音与视频信号的需求也越来越大，但是，当时的网络带宽还十分有限，而且音视频文件的体积一般都十分庞大。在这种情况下，看一个很短的视频片断就有可能花几十分钟甚至更长的时间等待，这不能不说是一件让人头疼的事，因而，急需一种技术来改善音视频传输困难的局面。这种技术就是流媒体技术。

流媒体又称流式媒体（Stream Media），是指在互联网上实时、流动地传播音、视频等多媒体内容的连续时基媒体，允许浏览者一边下载一边运行流媒体文件，流媒体技术并不是单一的技术，它是融合了网络技术之后所产生的技术。它需要涉及到流媒体数据的采

集、压缩、存储、传输以及网络通信等多项技术。

传统的播放多媒体方式是，先将整个媒体文件内容下载到本地，然后再播放。多媒体文件通常体积庞大，因此下载需要很长时间。而流媒体采用了流式传输技术，将多媒体文件经特定压缩方式处理成一个个压缩包，由视频服务器向用户计算机顺序或实时传送，用户不必等整个文件下载完毕，只需经很短的启动延时，即可利用计算机上的解压设备对文件第一部分进行解压播放。在第一部分开始播放的同时，文件的其他部分不断流出，一边下载一边播放，从而大大节约了时间。

注意： 如果将文件传输看作是一次接水的过程，过去的传输方式就像是对用户做了一个规定，必须等到一桶水接满才能使用它，这个等待的时间自然要受到水流量大小和桶的大小的影响。而流式传输则是，打开水龙头，等待一小会儿，水就会源源不断地流出来，而且可以随取随用，因此，不管水流量的大小，也不管桶的大小，用户都可以随时用上水。从这个意义上看，流媒体这个词是非常形象的。

9.1.2　流媒体的网络结构与技术原理

从结构上看，一个完整的流媒体网络结构，主要由流媒体发布服务器和流媒体网络终端组成，在不同的网络设备连接下，可接入不同的终端设备，如移动播放设备、显示终端、调音台、播放器等均可。一个普通的流媒体网络结构图如图 9.1 所示。

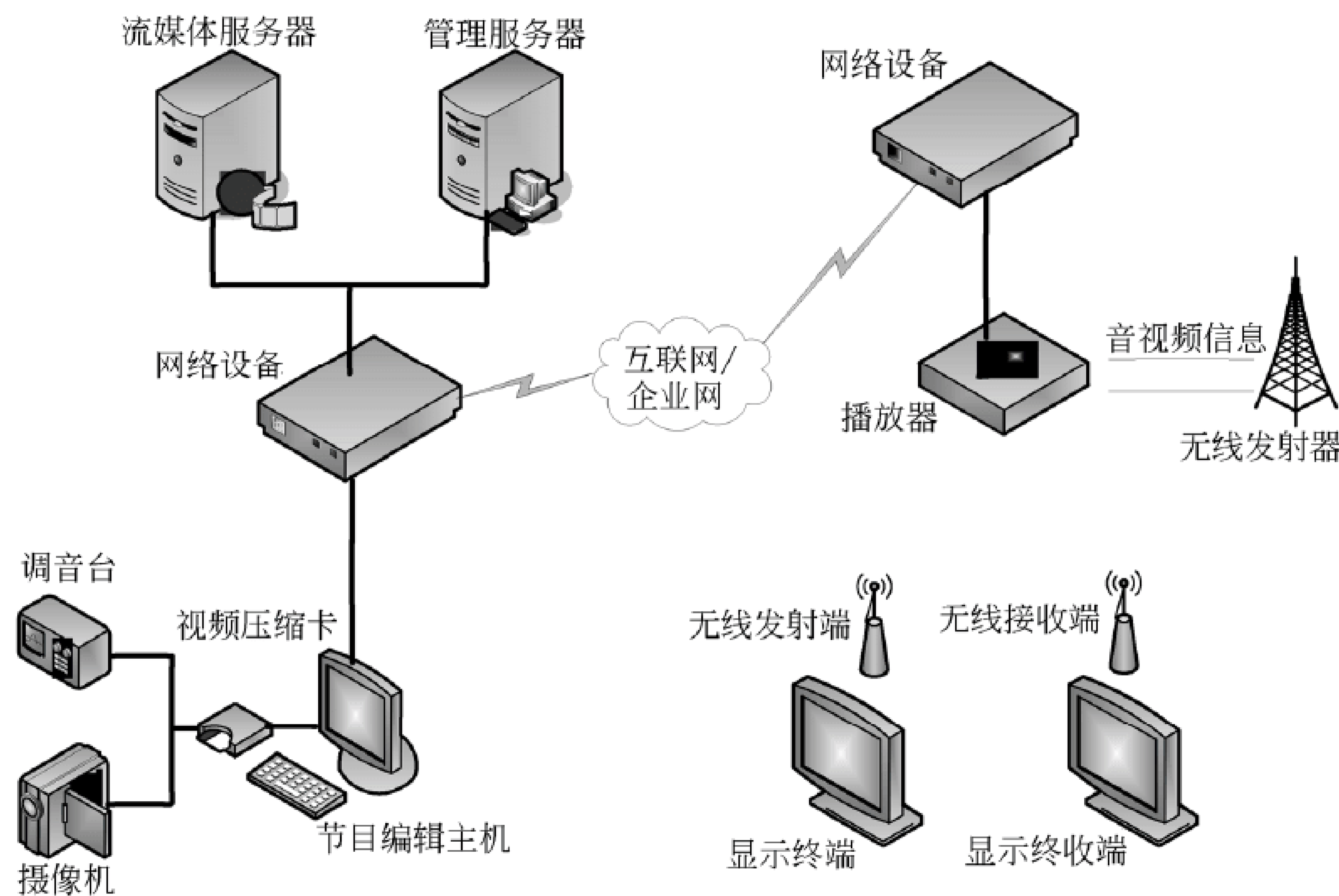


图 9.1　普通的流媒体网络结构图

在图 9.1 所示的基础上，要让整个流媒体系统工作起来，还需要一系列的处理流程，这个流程就是流媒体工作的基本原理。

流媒体技术工作原理的实现，需要一个完整媒体系统结构、需要缓存技术进行数据缓存、传输协议及合理的传输流程来发送和接收媒体数据等，它们共同构成了流媒体的基本原理。图 9.2 所示的就是流媒体的基本工作原理图。

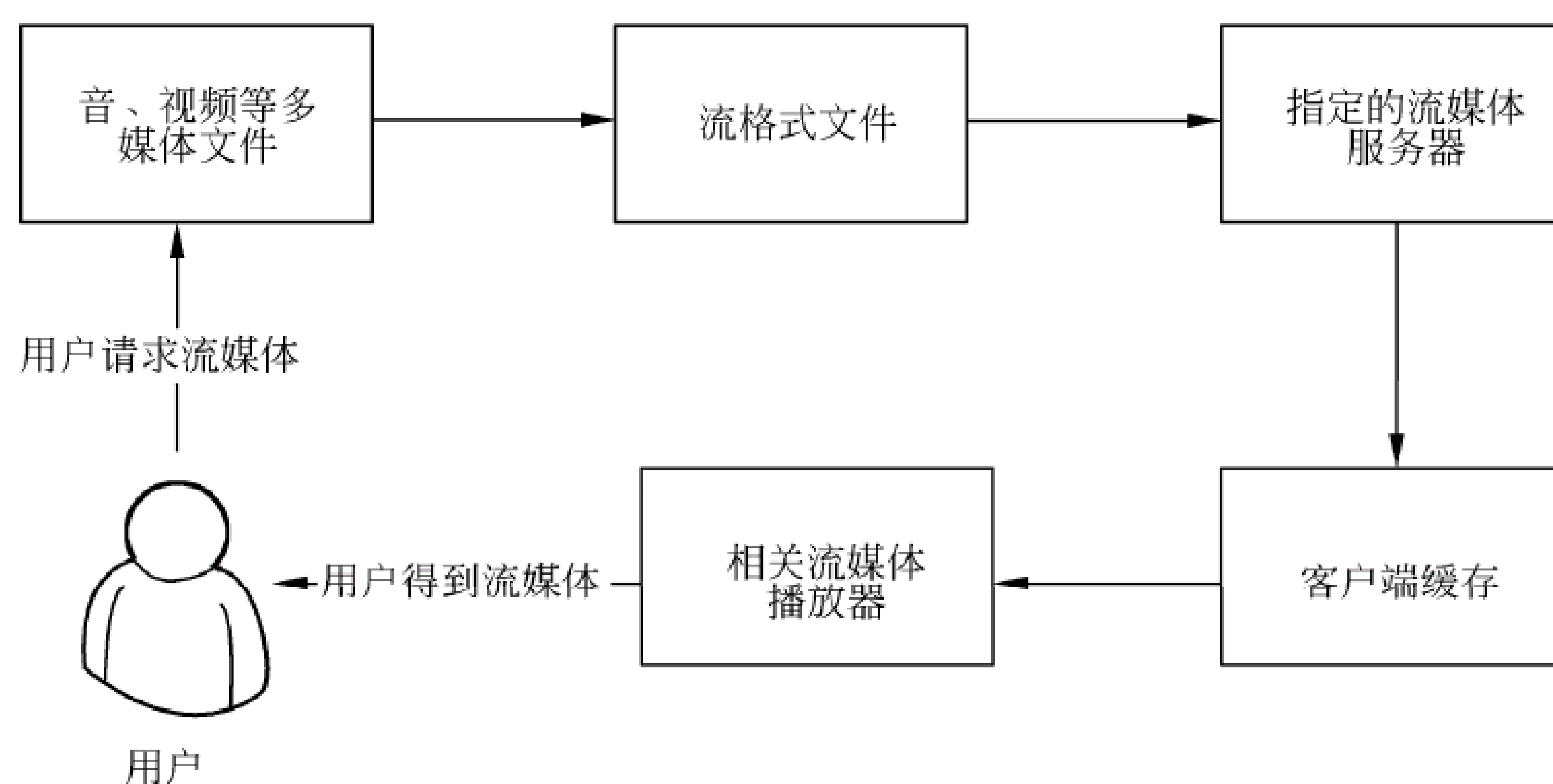


图 9.2　流媒体的基本工作原理

1．流媒体系统结构

现存流媒体解决方案采用的技术是多样的，但其体系结构的本质是相近的。流媒体的体系构成主要包括以下几个方面。

- 编码工具：用于创建、捕捉和编辑多媒体数据，形成流媒体格式。
- 流媒体数据：原始的流媒体源文件、包括音、视频源文件。
- 服务器：存放和控制流媒体的数据。
- 网络：适合多媒体传输协议甚至实时传输协议的网络。
- 播放器：供客户端浏览、运行流媒体文件软件系统，通常是独立的播放器和 ActiveX 方式的插件。

2．缓存技术

流式传输的实现需要缓存。因为一个实时音视频源或存储的音视频文件在传输中被分解为许多数据包，而网络又是动态变化的，各个数据包选择的路由可能不相同，故到达客户端的时延也就不同，甚至先发的数据包有可能后到。为此，需要使用缓存系统来消除时延和抖动的影响，以保证数据包顺序正确，从而使媒体数据能够连续输出。通常高速缓存所需容量并不大，因为通过丢弃已经播放的内容可以重新利用空出的空间来缓存后续尚未播放的内容。

注意： 时延，是指数据包第一个比特进入路由器到最后一比特从路由器输出的时间间隔。而时延抖动，指的就是时延的变化。

3．流媒体传输协议

流式媒体传输的实现需要与之相适应的传输协议，就 TCP 协议而言，需要较多的开销，故不太适合传输实时数据。在流式数据传输的实现方案中，一般采用 HTTP/TCP 来传输控制信息，而用实时传输协议 / 用户数据报协议（RTP / UDP）来传输实时流媒体数据。下面是几种常用的流媒体传输协议。

（1）实时传输协议 RTP 与 RTCP

RTP 是用于 Internet 中针对多媒体数据流的一种传输协议。RTP 被定义为在一对一或一对多传输的情况下工作，其目的是提供时间信息和实现流式数据同步。RTP 通常使用 UDP 来传送数据，但 RTP 也可以在 TCP 或 ATM 等其他协议上工作。

注意：ATM（Asynchronous Transfer Mode，异步传输模式），是一项数据传输技术。ATM 是以信元为基础的一种分组交换和复用技术，是一种为了多种业务设计的通用面向连接的传输模式。它适用于局域网和广域网，具有高速数据传输率和支持许多种类型如声音、数据、传真、实时视频、CD 质量音频和图像的通信。

RTCP 是针对 RTP 的控制协议（RTCP：RTP Control Protocol），此协议采用与数据包相同的分发机制，将控制包周期性传输到所有会话参与者中。底层协议必须提供数据和控制包的多路发送，RTCP 协议也提供数据分发质量反馈信息。

当应用程序开始一个 RTP 会话时将使用两个端口：一个给 RTP，一个给 RTCP。RTP 本身并不能为按顺序传送数据包提供可靠的传送机制，也不提供流量控制或拥塞控制，它依靠 RTCP 提供这些服务。RTCP 和 RTP 一起提供流量控制和拥塞控制服务。

RTP 和 RTCP 配合使用，它们能以有效的反馈和最小的开销使传输效率最佳化，因而特别适合传送网上的实时数据。

（2）实时流协议 RTSP

实时流协议 RTSP 是由 Real Networks 和 Netscape 共同提出的，该协议定义了一对多应用程序如何有效地通过 IP 网络传送多媒体数据。RTSP 在体系结构上位于 RTP 和 RTCP 之上，它使用 TCP 或 RTP 完成数据传输。

HTTP 与 RTSP 相比，HTTP 传送 HTML，而 RTP 传送的是多媒体数据。HTTP 请求由客户机发出，服务器做出响应；使用 RTSP 时，客户机和服务器都可以发出请求，即 RTSP 可以是双向的。

（3）资源预订协议 RSVP

由于音频和视频数据流比传统数据对网络的延时更敏感，要在网络中传输高质量的音频、视频信息，除带宽要求之外，还需其他更多的条件。RSVP 是 Internet 上的资源预订协议，使用 RSVP 预留一部分网络资源（即带宽），能在一定程度上为流媒体的传输提供 QoS。

4．流媒体的传输过程

流式媒体数据的传输主要是从流媒体服务器流向客户端，在传输的过程中有一套完善的流程，此过程如下。

- 用户选择某一流媒体服务后，Web 浏览器与 Web 服务器之间使用 HTTP / TCP 交换控制信息，以便需要传输的实时数据从原始信息中检索出来。
- Web 浏览器启动音视频客户程序，使用 HTTP 从 Web 服务器检索相关参数对音视频客户程序初始化，这些参数可能包括目录信息、音视频数据的编码类型或与音视频检索相关的服务器地址。
- 音视频客户程序及音视频服务器运行实时流协议，以交换音视频传输所需的控制信息，实时流协议提供执行播放、快进、快倒、暂停及录制等命令的方法。
- 音视频服务器使用 RTP / UDP 协议将音视频数据传输给音视频客户程序，一旦音

视频数据抵达客户端，音视频客户程序即可播放输出。

需要说明的是，在流式传输中，使用 RTP / UDP 和 RTSP / TCP 两种不同的通信协议与音视频服务器建立联系，目的是为了能够把服务器的输出重定向到一个非运行音视频客户程序的客户机的目的地址。另外，实现流式传输一般都需要专用服务器和播放器。

9.1.3　流媒体的传输技术

流媒体的传输，指的就是在互联网上将流媒体数据从一端传送到另一端，这一传输过程，主要就是通过 Internet 将包含影视节目信息的多媒体数据，从媒体服务器上传送到请求媒体数据的 PC 客户机上。如图 9.3 所示为流式数据传输的基本原理。

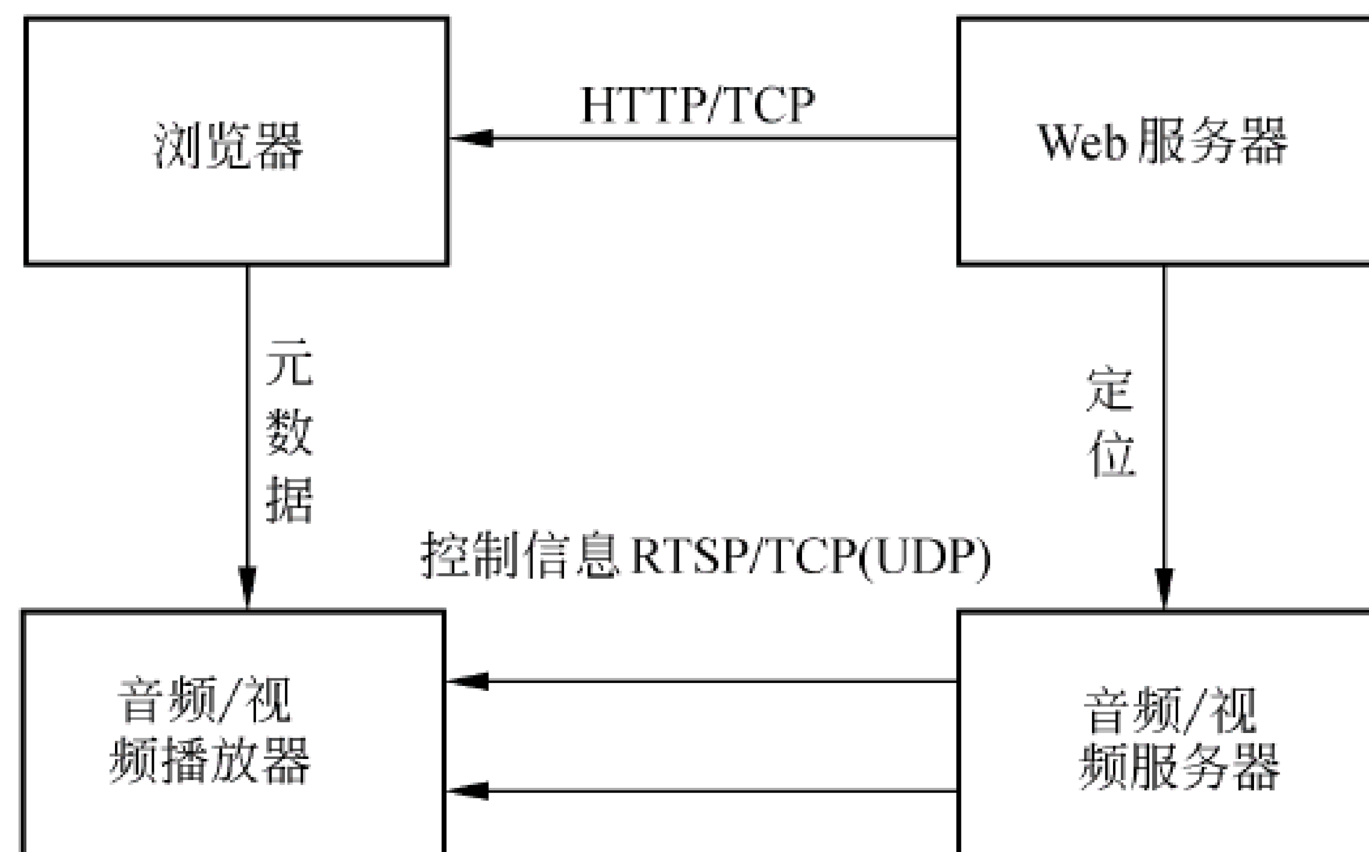


图 9.3　流式数据传输的基本原理图

实现流式传输有两种方法：实时流式传输（Real time streaming）和渐近流式传输（progressive streaming）。一般说来，如果流媒体数据为实时广播，或使用流式传输媒体服务器，或应用如 RTSP（Real Time Streaming Protocol）等实时协议，即为实时流式传输。如使用 HTTP 服务器，文件即通过渐近流发送。采用哪种传输方法取决于用户的需求。当然，流式文件也支持在播放前完全下载到硬盘。

注意： 渐近流式传输有的文件上也叫顺序流式传输。

（1）实时流式传输

在实时流式传输中，保证媒体信号带宽与网络连接匹配，音视频信息可被实时观看到。在观看过程中用户可快进或后退以观看前面或后面的内容，但是在这种传输方式中，如果网络传输状况不理想，则收到的信号效果比较差。

实时流式传输与 HTTP 流式传输不同，它需要专用的流媒体服务器与传输协议。

实时流式传输总是实时传送，特别适合现场事件，也支持随机访问，用户可快进或后退以观看前面或后面的内容。理论上，实时流一经播放就可不停止，但实际上可能发生周期暂停。

（2）渐进流式传输

渐进流式传输是顺序下载，在下载文件的同时用户可观看在线媒体，但是，用户的观看与服务器上的传输并不是同步进行的，用户是在一段时延后才能看到服务器上传出来的

信息，或者说用户看到的总是服务器在若干时间以前传出来的信息。在这过程中，用户只能观看已下载的那部分，而不能要求跳到还未下载的部分。

渐进流式传输不像实时流式传输在传输期间根据用户连接的速度做调整。由于标准的 HTTP 服务器可发送这种形式的文件，也不需要其他特殊协议，它经常被称作 HTTP 流式传输。

渐进流式传输比较适合高质量的短片段，如片头、片尾和广告，由于该文件在播放前观看的部分是无损下载的，这种方法保证电影播放的最终质量。这意味着用户在观看前，必须经历延迟，对较慢的连接尤其如此。

渐进流式文件是放在标准 HTTP 或 FTP 服务器上，易于管理，基本上与防火墙无关。渐进流式传输不适合长片段和有随机访问要求的视频，如讲座、演说与演示。它也不支持现场广播，严格说来，它是一种点播技术，适合于在网站上发布的供用户点播的音视频节目。

9.1.4　流媒体的文件类型

在运用流媒体技术时，音视频文件要采用相应的格式，不同格式的文件需要用不同的播放器软件来播放，所谓“一把钥匙开一把锁”。目前，采用流媒体技术的音视频文件主要有 3 大“流派”。

（1）一是微软的 ASF（Advanced Stream Format）。这类文件的后缀是.asf 和.wmv，与它对应的播放器是微软公司的 Media Player。用户可以将图形、声音和动画数据组合成一个 ASF 格式的文件，也可以将其他格式的视频和音频转换为 ASF 格式，而且用户还可以通过声卡和视频捕获卡将诸如麦克风、录像机等外设的数据保存为 ASF 格式。

（2）二是 Real Networks 公司的 Real Media，它包括 RealAudio、Real Video 和 Real Flash 三类文件，其中 RealAudio 用来传输接近 CD 音质的音频数据，Real Video 用来传输不间断的视频数据，Real Flash 则是 Real Networks 公司与 Macromedia 公司联合推出的一种高压缩比的动画格式，这类文件的后缀是.rm，文件对应的播放器是 RealPlayer。

（3）三是苹果公司的 QuickTime 和 iTunes。QuickTime 是由苹果计算机所开发的一种多媒体架构，能够处理许多的数字视频、媒体段落、音效、文字、动画、音乐格式，以及交互式全景影像的数项类型，iTunes 是 Apple 公司的另一款媒体播放软件，新版的 iTunes8 目前可以读取、写入以及转换于 MP3、AIFF、WAV、MPEG-4、AAC 与 Apple Lossless 之间。

注意： iTunes 是一款媒体播放器的应用程序，2001 年 1 月 10 日由苹果电脑在旧金山的 Macworld Expo 推出，用来播放以及管理数位音乐和与视讯档案，依然是管理苹果电脑最受欢迎的 iPod 的档案的主要工具。此外，iTunes 能连线到 iTunes Store（在有网络连线的情况下），以便下载购买的数位音乐、音乐视讯、电视节目、iPod 游戏、各种 Podcast 以及标准长片。

总结以上 3 家公司，它们所支持的流媒体文件格式、文件类型，如表 9.1 所示。

表 9.1　流媒体文件类型及文件格式

公　　司	文 件 格 式	媒 体 类 型
微软	ASF（Advanced Stream Format）	Video/x-ms-asf
Real Networks	RM（Real Video）	Application/x-pn-realmedia
	RA（Real Audio）	Audio/x-pn-realaudio
	RP（Real Pix）	Image/vnd。rn-realpix
	RT（Real Text）	Text/vnd。rn-realtext
Apple	MOV（QuickTime Movie）	Video/quicktime
	QT（QuickTime Movie）	Video/quicktime

以上的这些文件类型，基本上占据了流媒体主流的应用，如图 9.4 所示，就是针对这 4 种主流的媒体播放器的使用情况统计。

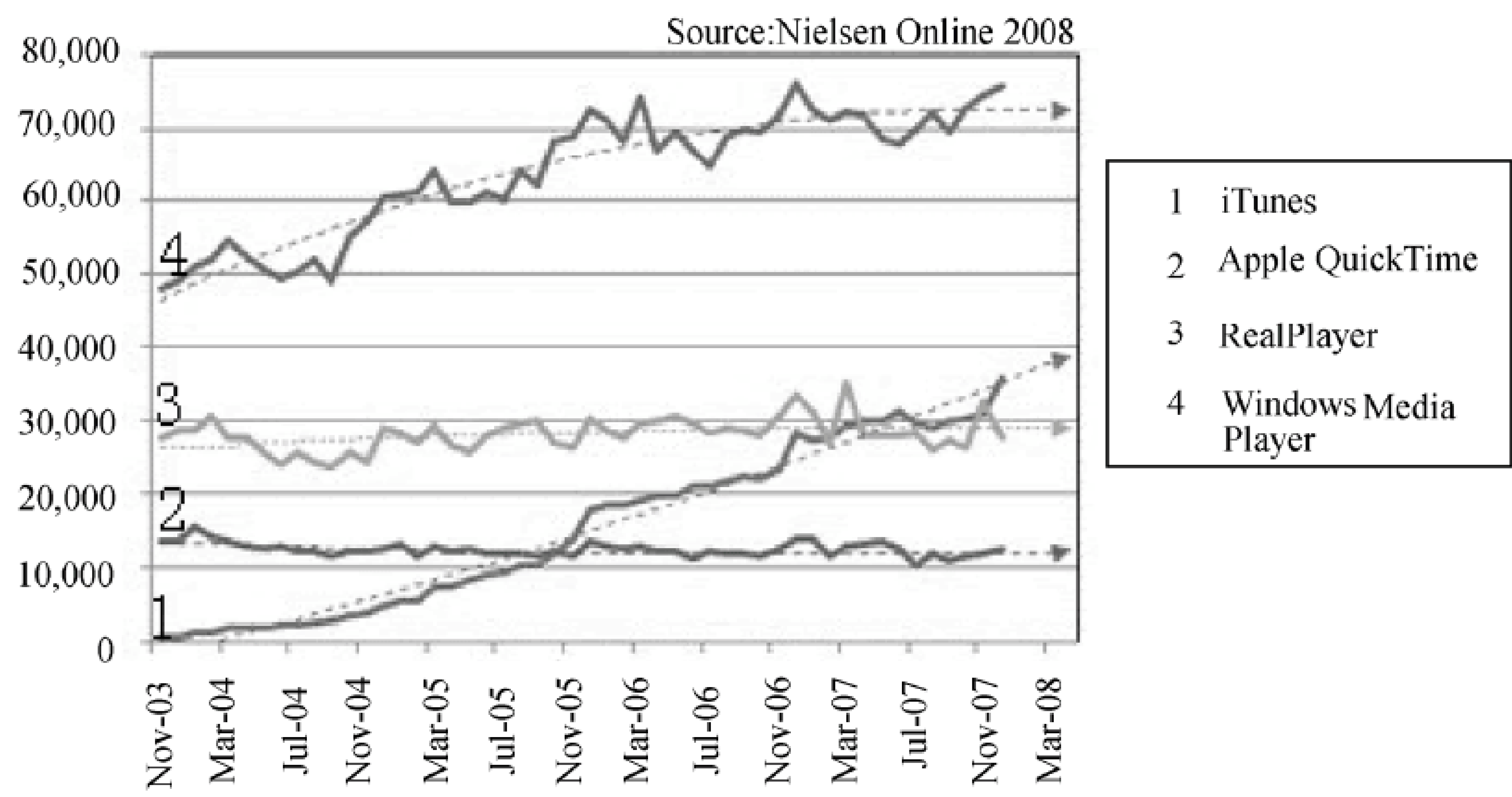


图 9.4　主流的媒体播放器用户使用情况统计图

图 9.4 所示的是：从 2003 年 11 月到 2007 年 12 月，每 1000 个人中，使用这 4 种媒体播放器的人数走向曲线图。可以看出，微软的 Windows Media Player 占有绝大多数的用户，它和 Apple 的 iTunes 都呈增长趋势。

在应用流媒体的时候，除了这些常用的文件格式类型，还有一些发布文件，例如 RAM、ASX，这类文件本身就不是影音文件，它们的作用在于给出真正流媒体文件所在的位置，其实这个文件在流媒体播放的过程中不是必需的。如表 9.2 所示为一些常用的流媒体发布文件的格式。

表 9.2　常用的流媒体发布文件的格式

流媒体发布文件格式	注　　释
ASX	Active Stream Redirector
RAM	Real Audio Media
RPM	Embedded Ram
SMI/SMIL	Synchronized Multimedia Integration Language
XML	Extensible Markup Language

9.1.5　流媒体的应用及发展

由于流媒体技术在一定程度上突破了网络带宽对多媒体信息传输的限制，因此被广泛运用于网上直播、网络广告、视频点播、远程教育、远程医疗、视频会议、企业培训、电子商务等多种领域。目前应用最直接的是网上直播。作为新一代互联网的标志，宽带流媒体彻底改变了传统互联。

总地来说，流媒体技术改变了互联网只能表现文字和图片的缺陷，实现了一种可集音频、视频及图文于一体技术。流媒体将成为未来互联网应用的主流，并将推动互联网整体架构的革新。

根据 Yankee Group 发布的研究报告，家庭宽带接入的快速发展将推动流媒体广告的繁荣，预计到 2005 年，美国的流媒体广告市场将达到 31 亿美元，2000 年流媒体市场仅为 4400 万美元，足可风流媒体巨大的价值。

微软董事长比尔・盖茨曾说过：“流媒体是微软真正看好的方向”。而雅虎公司的创始人之一杨致远在 2000 年“西方流媒体技术会议”上发表演讲时指出，流媒体技术的发展已经达到了从量变到质变的关键时刻。的确如此，伴随着宽带网络用户数量日益增加，流媒体的时代已经到来了。

在中国，也有专业机构对中国流媒体业务的发展情况进行了预测，预测的走势图如图 9.5 所示。

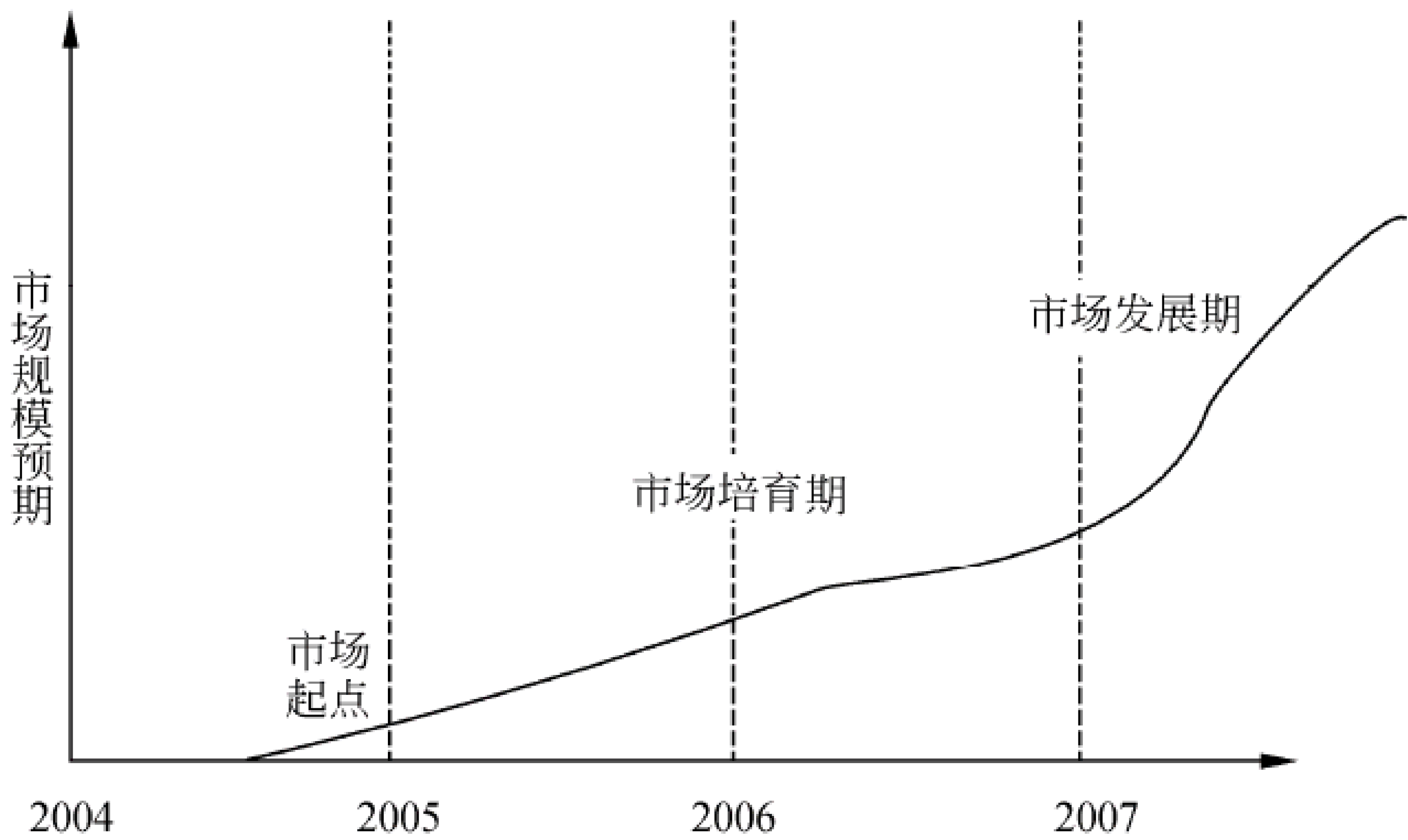


图 9.5　流媒体业务在中国的发展阶段预测图

如图 9.5 所示的，2005 年以前，国内流媒体业务发展还面临诸多障碍，属于发展的走步阶段。但随着产业模式的不断推进、产业链的逐步完善和商务模式的逐渐成熟，以及各种外部的制约因素逐步得到解决，到 2007 年的时候国内的流媒体业务必迎来大规模的发展，事实也证明了这种预测，当前流媒体业务在国内发展迅速，流媒体技术在各方面也都得到广泛的应用。

发挥传统媒体的优势，利用网络媒体的特长，保持媒体间良好的竞争与合作，是未来网络的发展之路，也是未来传统媒体的发展之路。流媒体技术的运用只是一个开端，无数

新的技术还在前面等着我们。

9.1.6　流媒体技术面临的问题

尽管流媒体技术呈爆炸式增长，但还必须面对不少障碍，必须看到，目前流媒体平台在数据分发、数据流传输、服务器管理等方面还面临许多问题。

传统流媒体服务都是 C/S 模式，即用户从流媒体服务器点击观看节目，然后流媒体服务器以单播方式把媒体流推送给用户。这种 C/S 模式加单播方式的缺陷在流媒体业务发展到一定阶段。用户数量上升后就会显现出来。主要有下面两个问题。

（1）流媒体服务器带宽占用大。不同于电台和电视台使用广播形式发送节目流媒体业务使用了单播形式，即一个用户一份流，即使有两个用户在观看同一个节目。所以用户数和服务器带宽消耗是成正比的关系，用户越多。流媒体服务器需要的带宽就越多。

当用户到达一定规模后。带宽就会成为业务发展的瓶颈，这时就需要投入大量费用购买带宽以满足要求。同时，流媒体服务器处理能力也要求很高。大量的用户登录流媒体服务器后，必然要求服务器有较高的处理能力，这时候就需要更高性能的服务器以支持更多用户，这无疑会增加成本

（2）流媒体负载均衡要求高。为减少骨干网络带宽占用、保证服务质量和就近提供服务，一般流媒体服务都需要部署复杂的内容分发系统（CDN）。这样就大大增加了系统投资和管理复杂度。

注意： CDN 的全称是 Content Delivery Network，即内容分发网络。其目的是通过在现有的 Internet 中增加一层新的网络架构，将网站的内容发布到最接近用户的网络“边缘”，使用户可以就近取得所需的内容，解决 Internet 网络拥挤的状况，提高用户访问网站的响应速度。从技术上全面解决由于网络带宽小、用户访问量大、网点分布不均等原因所造成的用户访问网站响应速度慢的问题。

流媒体业务在到达一定阶段后。就需要大规模扩充带宽、服务器和内容分发系统以满足需求这些举措无疑都会大大增加开销。同时还要随着用户量的增加、媒体数量的增加而不断扩容。

这种一边扩容一边发展用户，扩容完了再大力发展用户的模式是实质上是一种面多了掺水，水多了掺面的思路，这只是一种权宜之计。它无法从根本上解决流媒体业务发展所遭遇到的瓶颈问题。

在这种背景下，P2P 技术开始走入人们的视野。一些厂商开始尝试在流媒体领域引入业界先进的 P2P 技术。P2P 流媒体技术和传统流媒体不同之处在于用户在播放过程中不仅仅可以从流媒体服务器取得媒体数据流，还可以从其他的对等用户那里取得媒体数据流。与此同时，用户还会向其他用户提供自己拥有的、别人需要的媒体流。

通过引入 P2P 技术，可以很好地解决流媒体服务器占用带宽、负载过重的情况，后来的事实证明，P2P 与流媒体技术的完美结合，再一次带来整个互联网中流媒体业务的辉煌发展。

9.2　P2P 与流媒体技术的结合

P2P 在设计之初，主要是用来于网络文件共享、信息交换的技术，并且已经在这些方面 P2P 有过杰出的表现。然而随着互联网多媒体产业的发展，以及流媒体所面临的问题，P2P 再一次在流媒体领域担起重任，要是因为 P2P 技术解决了流媒体（网络电视）技术的众多难题。

9.2.1　P2P 技术应用到流媒体中的特性

P2P 网络是一个对等连接的网络，是指不同系统之间通过直接交换，实现计算机资源和服务共享的一种应用模式。P2P 使得网络上的沟通变得容易、更直接共享和交互。

从 P2P 应用到流媒体技术中来看，它有两个重要的特点：

- P2P 改变了互联网现在的以大网站为中心的状态、重返“非中心化”，并把权力交还给用户。
- 在 P2P 系统中，每一个 Peer 都是平等的参与者，承担服务使用者和服务提供者两个角色。

P2P 技术所具备的这种特性使得流媒体的发布不再以“大服务器”为中心，资源的所有权和控制权被分散到网络的每一个结点中。服务使用者和服务提供者之间进行直接通信，这样，可充分利用网络带宽，减少网络的拥塞状况，使得资源的有效利用率大大提高。

利用 P2P 的这些特性后，在一个流媒体系统中，每个流媒体用户就是一个 P2P 网络中的一个结点，这些结点是通过 P2P 网络组织起来的，它们就都符合 P2P 技术中对等、非中心化的特性。一个对等结点的子集拥有一个特定的媒体文件（或文件的一部分），并为所有的需要访问此文件的其他结点提供媒体数据。与此同时，请求数据的结点在下载媒体数据的过程中回放并存储这个媒体的数据，并成为可以为其他结点提供流媒体数据上载的结点。如图 9.6 所示为流媒体系统中的 P2P 传输示意图。

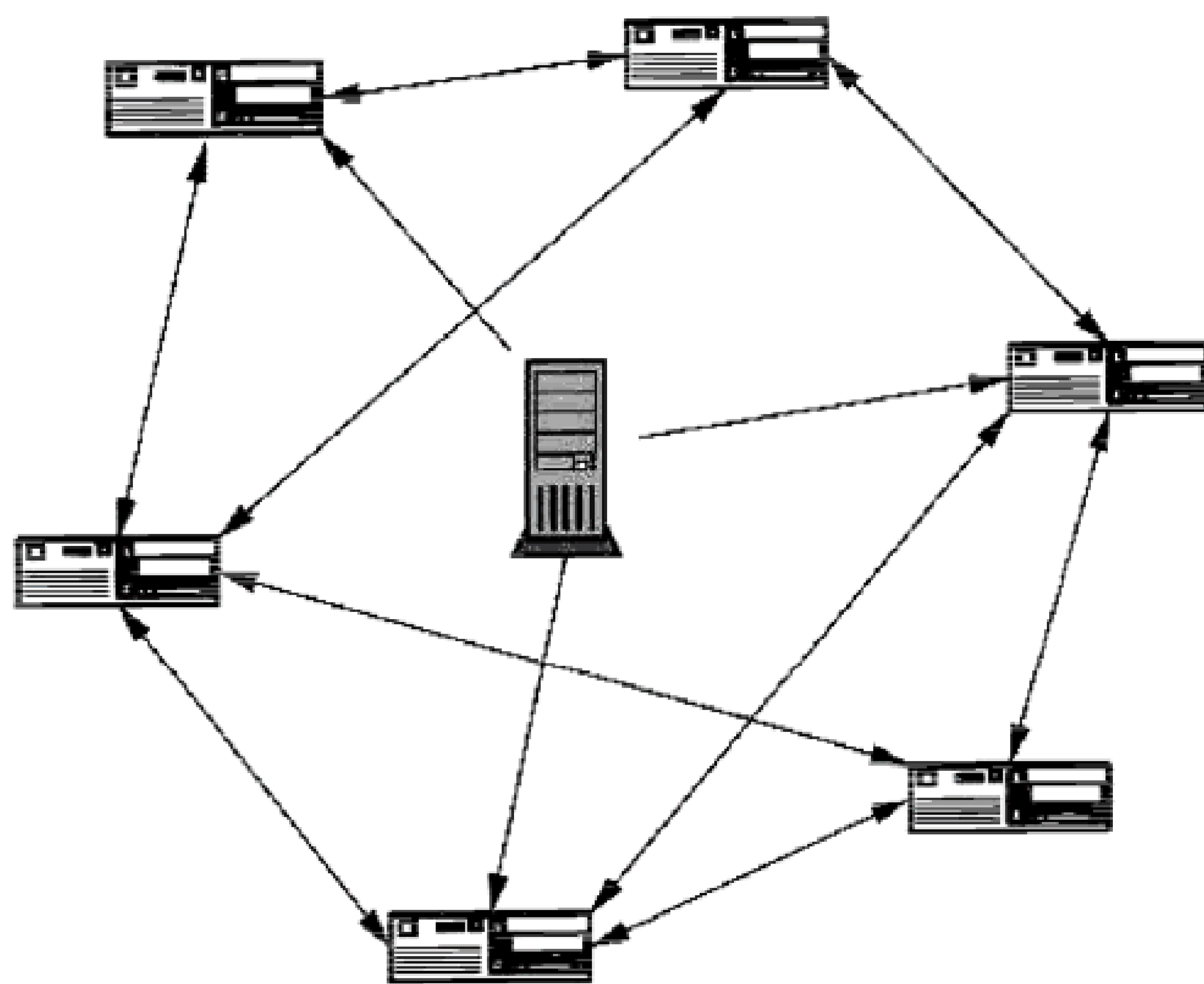


图 9.6　P2P 传输示意图

所以，使用流媒体的用户就可以根据他们的网络状态和设备能力与一个或几个用户建立连接来分享数据，这种连接，不再是以往只与服务器的连接，而是连接自己的对等 Peer。而且建立连接以后，在下载媒体数据的同时也上传数据，这样，用户数越多，上传的数据量也就越大，下载的来源就更广。所以，看 P2P 网络电视，同一个节目，看的人越多、节目播出就越流畅，也就是这个原理。

引入 P2P 技术以后，流媒体的分发能有效地减少服务器的负担和提高每个用户的视频质量。同时，P2P 技术在流媒体应用中特别适用于一些热门事件，即使是大量的用户同时访问流媒体服务器，也不会造成服务器因负载过重而瘫痪。

基于 P2P 的流媒体服务系统并不改变现有的流媒体服务架构，只是在现有系统的基础上，改变传统模式下的服务方式和数据传输路径，使请求同一媒体流的客户端组成一个 P2P 网络，使服务器只须向这个 P2P 网络中的少数结点发送数据。而这些结点可以把得到的数据共享给其他的结点，每个结点依然可以通过流媒体系统得到高质量的视频服务。

总地来说，基于 P2P 的流媒体技术为解决流媒体内容分发提供了一个新的方向，逐渐引起了人们的重视。许多大学、研究机构、公司等都对此进行了研究并开发出相关技术或原型系统。

9.2.2　P2P 与流媒体的完美结合

P2P 技术与流媒体的完美结合，可以有效地解决流媒体技术发展上的瓶颈问题，基于 P2P 技术的流媒体发布与传统流媒体发布技术相比较而言，具有下列优势：

P2P 系统不同于传统的 C/S 工作方式，使其具有了新的特点：

1．流服务能力的提高

- P2P 流媒体传输的内容与传统的流媒体发布的内容有所不同，在核心结点根据 P2P 协议对内容（包括文件和流）做切片处理，P2P 用户将根据这些规则来完成 P2P 共享。P2P 在边缘层的引入大大降低了边缘服务器的压力，提高了文件传输和流媒体传输的效率。
- P2P 流媒体技术充分利用了用户的闲置上行带宽，这样运营商可以通过更少的边缘服务器，提供更多的业务量为更多的用户服务，以较低成本代价应对迅猛增长的客户规模带来的挑战。

注意：试验证明，P2P 流媒体在千人规模的情况下播放 500K 码流左右的节目，服务端 CPU 消耗小、带宽消耗小，而且用户服务质量高。

2．客户体验的改善

- P2P 和流媒体结合的方式，使得有限的服务能力可以为更多的用户提供流媒体服务。另一方面，P2P 技术的应用也能够更有效地防止因网络的抖动而产生对服务质量的影响。
- 内容的丰富是 P2P 流媒体一大特点，直播、点播、录播等方式种类齐全。不仅央视、省级卫视的众多电视频道可以实时、延时收看，还可提供其他经典节目的

点播。

- 互动性强，基于 P2P 的流媒体，观众可以不受时间、频道、内容的限制，随时点播、观看、录制所需要的节目内容。

3．可以有效拓展宽带网络增值业务

- 大大减少了流媒体服务提供商的网络及服务器硬件投资，降低了业务进入的门槛。
- 在不对现有流媒体平台进行根本性变动情况下，极大地提高了服务用户数量，甚至大量用户接入成为优势。
- 有效提升了宽带网络的利用率。从长远看这种新的业务模式有可能将成为促使骨干网运营商考虑升级到更快、更先进的网络架构的因素。
- 顺应互联网技术发展的大趋势。应用成果不仅可用于流媒体平台，还可以拓展到其他互联网增值业务领域，有利于进一步进行其他新业务的研究。

P2P 技术不仅可以有效地解决传统流媒体技术中的一些问题，而且 P2P 技术以其结点数量大、动态性强、异构性强、分布广泛、网络异步性强等特点又为流媒体技术的发展带来了新的研究空间。一个成熟的 P2P 的流媒体技术需要在不断的实践、不断的探索前进。

9.2.3　P2P 流媒体技术的研究情况

从整个流媒体的发布形式上看，总地来说它存在着直播与点播种这两种发布形式，而由于直播与点播各自的特性要求，使用它们几乎没有太多可研究的共性可言，所以，从流媒体的整个发展来看，P2P 流媒体技术仍然是按照直播、点播这两个分支来分别研究的。

1．P2P流媒体的分类

广播电视词典对直播界定为“广播电视节目的后期合成、播出同时进行的播出方式”。按播出场合“可分为现场直播、播音室和演播室直播”。

目前，网络媒体自身还没有给出准确的关于直播的定义，为了方便起见，不妨参照传播学及电视现场直播的概念给网络直播下个简单的定义：在现场随着事件的发生、发展进程同步制作和发布信息，具有双向流通过程的信息网络发布方式。直播，最大的特点就是实时性，必须在最短的时间内，把直播现场正在发生的情况准确无误的实时传递给用户。

直播服务对交互性要求很低，流媒体数据没有明确的文件头尾，用户加入后只能从流的当前位置开始收看，无法控制播放的进行度。正是由于直播服务对交互性的较低要求，技术实现与点播服务相比较为简单，使得 P2P 流媒体直播技术发展较为迅速也相对成熟。

视频点播的英文是 Video On Demond，缩写 VOD，是根据用户需要播放相应视频节目的一种新型服务。在 P2P 流媒体点播服务中，播放的媒体数据是已经录制好的、存储在固定服务器上的内容，对实时性要求不高，但必须要有足够快的响应速度，用户可以对点播的视频节目施加各种控制，如停止、暂停、重新开始、快进、快退等。因而其实现的复杂程度远远高于直播服务。

视频点播业务的出现，从根本上改变了用户过去被动式看节目的不足，在需求的推动下，P2P 点播视频技术也得到的较快的发展，目前已有多款成熟的基于 P2P 技术的点播系统平台。

2．P2P直播技术

P2P 流媒体直播服务，基本都采用了应用层组播技术，系统中的所有结点组成应用层覆盖网，并利用结点的能力实现流媒体数据分发功能。按照应用层覆盖网中结点的组织方式，P2P 流媒体直播技术分为 3 类，树型、网型和数据驱动型。

（1）树型组织方式

在树型的组织方式下，整个覆盖网络形成一棵以数据源为根的组播树。新加入结点的请求沿着树根向下遍历，直至找到一个拥有足够能力的结点，加入请求被重定向至该结点，由该结点为其服务。如图 9.7 所示，就是 P2P 直播技术的树型组织方式。

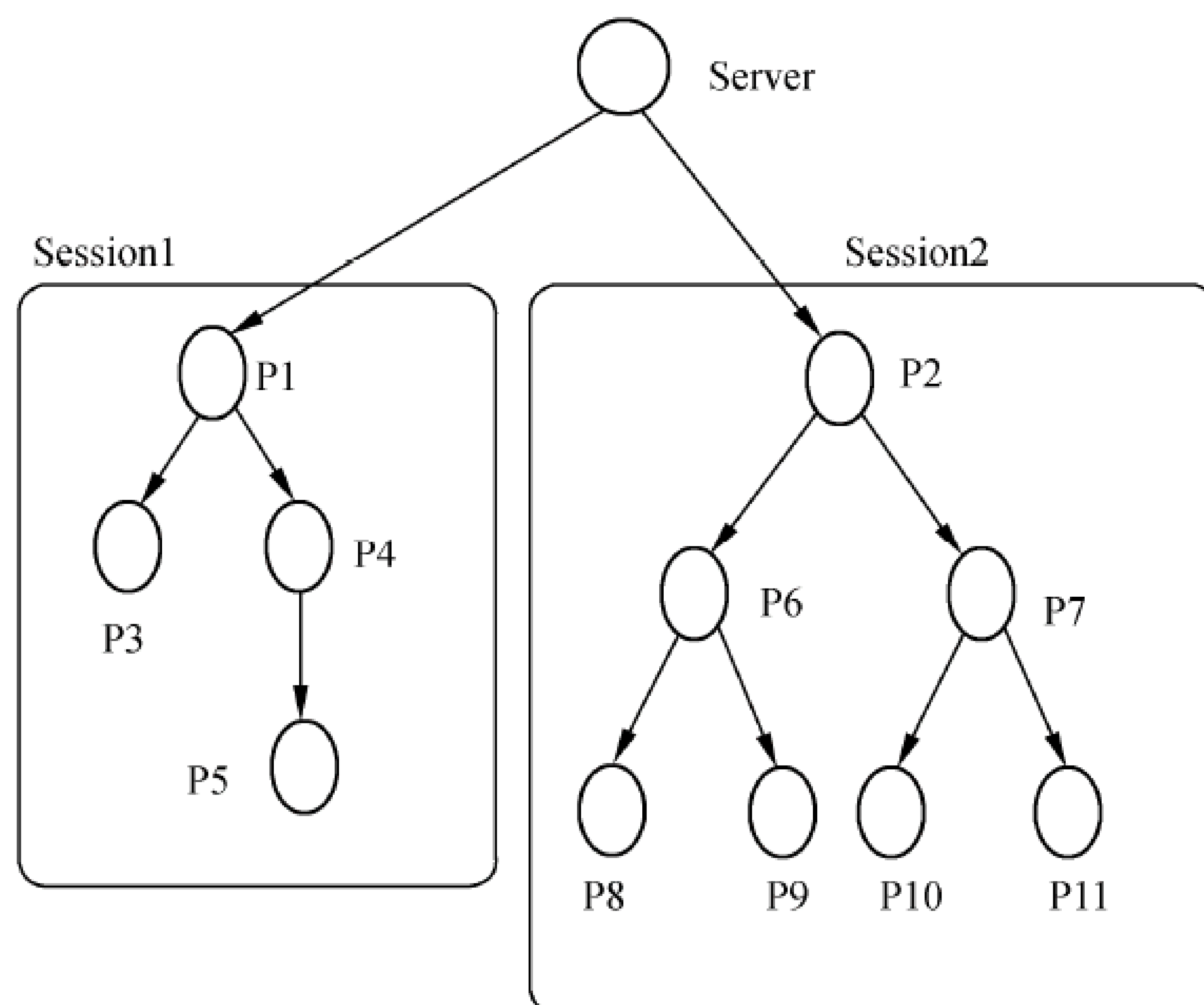


图 9.7　P2P 直播技术中的树型组织方式

树型结构的控制开销较小，一旦组播树建立后，父结点只需将收到的媒体数据转发到各个子结点。但是，这种结构存着很多问题，如没有考虑结点的异构性、容易单点失效、易形成负载不均等。

（2）网型结构

在网型结构方式下，所有结点首先在应用层构建了一个网状的覆盖网，然后基于覆盖网按照某种路由协议生成组播路由树，并通过路由树分发数据。

网型结构很好地解决了单点失效问题，系统鲁棒性与稳定性有所提高，同时对结点异构性也进行了一定的考虑。但是在该结构中，必须维护任意两个结点间的连接，即每个结点必须保存其余所有结点的列表，控制与维护开销随着系统中结点数的增加呈指数级增长，从而导致了网络效率低下，可扩展性极差，使得这种结构只能应用于小规模的网络中。

（3）数据驱动型

与树型结构和网型结构不同，在数据驱动的方式下，体系结构不是固定的，而是随着数据的分发而动态变化。所有结点周期性地交换各自所拥有的流媒体数据的信息，并根据接收到的信息与需要的流媒体数据，决定自己的邻居结点。一旦确定邻居结点以后，将从

多个邻居结点处获取数据。

数据驱动型结构方式不需要在结点之间交换复杂的控制信息，并且能够根据其他结点所拥有的流媒体数据的信息动态地确定邻居结点的位置并与它们交互。这种结构不但成功解决了单点失效问题，保证了系统的鲁棒性与稳定性，而且具有良好的可扩展性，同时还有效地解决了结点异构性问题。

注意：数据驱动的覆盖网络与基于树结构的最大不同在于它不组建和维护一个传输数据的明显拓扑结构，它用数据的可用性去引导数据流，而并不是在高度动态的 P2P 环境下不断地修复拓扑结构。

3．P2P点播技术

与直播技术不同，P2P 点播技术的发展速度相对缓慢，一方面是因为点播当中的高度交互性，在实现的时候复杂程度较高；另一方面是节目源版权因素对 P2P 点播技术的阻碍。目前，P2P 的点播技术主要朝着适用于点播的应用层传输协议技术、底层编码技术，以及数字版权技术等方面发展。

P2P 流媒体点播技术需要考虑用户对点播视频的各种控制操作，因此仅仅采用应用层组播技术无法满足点播的需求。当前，P2P 视频的点播技术解决方案主要用两种，一种是 P2Cast 系统，另一种是 PROMISE 系统。

P2Cast 系统，采用了应用层组播的方法，所有结点组成了一棵以数据源为根的组播树。同时，为了减轻对数据源服务器的负载，它引进了补丁（patching）机制作为应用层组播的补充。但是，P2Cast 不支持用户对点播视频节目的控制操作，新加入的用户结点必须从流的开始位置收看。从某种意义上说，它并不是纯粹的 P2P 视频点播系统

另一个是 PROMISE 系统，在 PROMISE 系统中，可以基于任意 P2P 协议（Pastry，Chord，CAN 等）构建应用层覆盖网，结点通过分布哈希表（Distributed Hash Table，DHT）查找获得一个拥有所请求文件的结点列表，然后通过特定的策略选择其中最好的 n 个结点作为父结点，其余的作为备用结点，以应对结点突然离开的状况。

该模型详细提出了最优结点的选择策略，传输分配策略以及动态调整策略。其中选择结点采用了基于拓扑感知的方法，不但考虑到结点本身的性能和带宽，还考虑到传输的路由（是否有结点传输的路由重叠），并且实时监控结点的变化，动态调整父结点集，使得传输的性能最优化。不足的是 Promise 的算法复杂性较高，且随着结点个数的曾加成指数上涨。

关于 PROMISE 系统其具体算法的详细内容，读者可自行查阅相关资料，这里就不再详述。

注意：关于 Pastry、Chord、CAN 这几种 P2P 协议，读者请参考本书 P2P 网络结构等章节的内容。

9.2.4　P2P 流媒体技术的意义及应用前景

P2P 技术与流媒体的结合，让流媒体运营商看到了巨大的商机，在商业运作的推动下，

P2P 流媒体业务得到了快速的发展。在网络视频领域以 P2P 技术为主的流媒体平台渐渐成为主流，给人们的网络生活、整个互联网的应用结构都带来了巨大的影响。

1．P2P流媒体的意义

P2P 流媒体技术能有效缓解服务器压力。P2P 流媒体技术可以有效利用网民的空余带宽，大大降低流媒体服务器压力。从而在同等条件下支持到更多的流媒体用户。对于流媒体业务发展具有重要意义。

P2P 技术可以提高用户收视质量。因为用户可以根据网络延时、响应速度等参数选择较快的相邻结点进行连接。从而避免了传统流媒体方式下单一地从局端服务器获取数据的方式。从这个意义上说，P2P 技术能够就近、就快地服务用户，从而大大提高了流媒体服务的质量。

2．P2P流媒体技术的应用现状及展望

目前在互联网世界，基于 P2P 的流媒体技术已经得到了广泛的应用，各种各样的产品层出不穷。目前比较流行的基于 P2P 的播放平台如 PPLive、PPStream、QQLive 等，它们都取得了非常大的成功，在赚取巨额利润的同时也推动着 P2P 视频技术不断向前发展，如图 9.8 所示的就是 P2P 流媒体市场的广告收入规模。

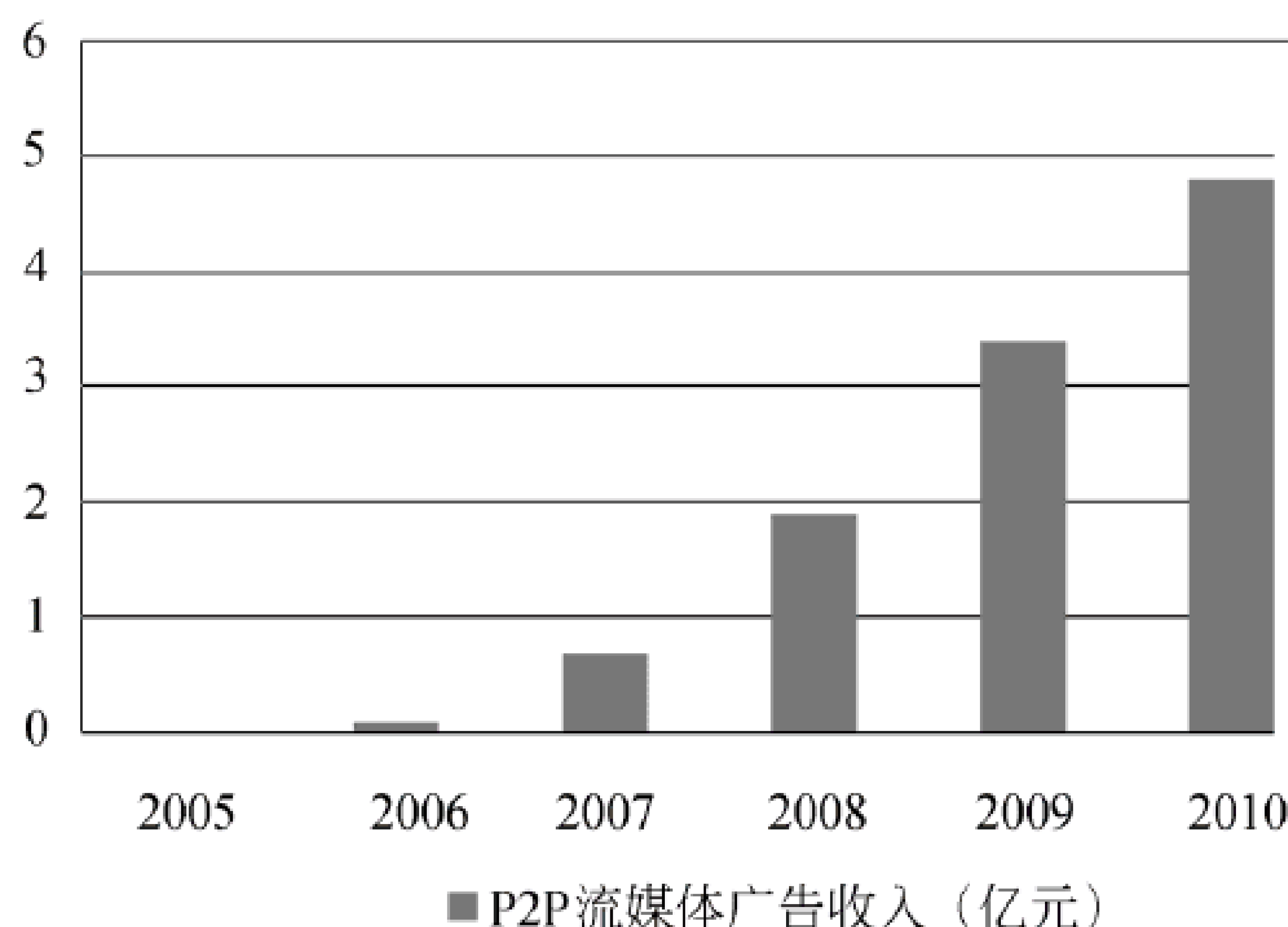


图 9.8　2005 年～2010 年中国 P2P 流媒体市场广告收入规模

图 9.8 就能充分说明了基于 P2P 的流媒体技术在应用上有着非常大的前景，不论是从其广告业务还是其他的盈利模式中，都可谓“钱”途光明。

当然，P2P 的流媒体技术发展至今，还存在着不少问题，未来的 P2P 流媒体发展还可以在以下方面有所突破。

（1）建立良好的安全机制

目前的 P2P 系统是没有安全保障的，一些结点可能通过散播恶意信息或垃圾信息来破坏整个系统。在不影响系统性能的前提下加入良好的安全机制，是流媒体未来的一个重要的研究方向。

（2）形成一个完美的激励机制

P2P 的特点在于共享，结点间的互利，在一些 P2P 网络中，上传下载等共享策略是随用户的意愿可变的，用户的个体选择导致客观的 P2P 网络呈现一定的社会性。个体用户的

行为，控制上传使得系统的共享性容易受到影响，因此，要保证 P2P 流媒体网络的健壮与流畅，应该进一步完善激励机制，鼓励共享，充分营造 P2P 中“人人为我，我为人人”的理念。这也是 P2P 流媒体进一步发展的重要因素。

（3）建立一个更加纯粹的 P2P 流媒体网络

当前的 P2P 流媒体系统无法真正做到类似于 eMule 那样的共享机制，用户所接收到视频信息都来源于服务提供商，而无法将自己本地的视频资源加入到流媒体网络中，这也是 P2P 流媒体技术可以研究的一个方面。

（4）移动流媒体业务

未来的 P2P 流媒体终端可能不仅仅是电脑，随着 3G 时代的到来，更多的移动终端将是 P2P 流媒体的潜力市场，这也会是未来的热点问题。

总地来说，当前的基于 P2P 技术的流媒体应用正处于一个快速发展的阶段，在未来网络电视、无线流媒体、数字家庭等应用领域，也必将有 P2P 流媒体技术的身影。所以，学好这门技术有着实用的意义和价值。

9.3　P2P 流媒体关键技术分析

基于 P2P 的流媒体技术，是一套完整的技术体系，包括多方面的内容，其应用的关键技术主要有流媒体的数据分发策略、数据处理与编码技术、媒体文件的定位技术、数据同步与拓扑均衡、数据交换技术、数据缓存技术及流媒体系统中的一些重要机制，下面分别对这些技术进行简单的说明。

9.3.1　流媒体的数据分发策略

目前，基于 P2P 网络的流媒体数据分发策略大致可以划分为 3 类，分别为以树结构为基础的分发策略，以森林（多树）结构为基础的分发策略，以及网状结构为基础的分发策略。下面分别简要介绍这 3 种策略方法。

1．单组播树结构

单组播树的结构，指的是基于 P2P 的流媒体网络中，将播放同一频道节目的所有结点组成一棵树的结构，提供媒体数据的源结点为这棵树的根。其他的结点依次为这个节目的孩子，这棵树中的每个结点可以为下层几个结点提供数据。

在初期的视频流媒体分发方案中，多采用基于组播树的拓扑结构，其典型的代表系统有 AigZag、PeerCast 等。如图 9.9 所示为单组播树的数据分发示意图。

在单组播树的结构中，每个树中的非叶子结点从自己唯一的父结点得到全部数据，再复制转发给自己所有的子结点，叶结点只从父结点得到数据，不再复制转发。当组播树中的非叶子结点退出时，它的子结点将暂时得不到数据，这时，系统需要尽快重建连接，保证所有结点都在组播树中。单组播对的这种架构对上层结点的依赖性过大，在结点的动态加入和退出的情况下，树结构不易维护，一旦在短时间内结点数量过多，会对上层结点造成过大的负担。另外，还存在传输延迟的问题，所以树的高度不能太大。

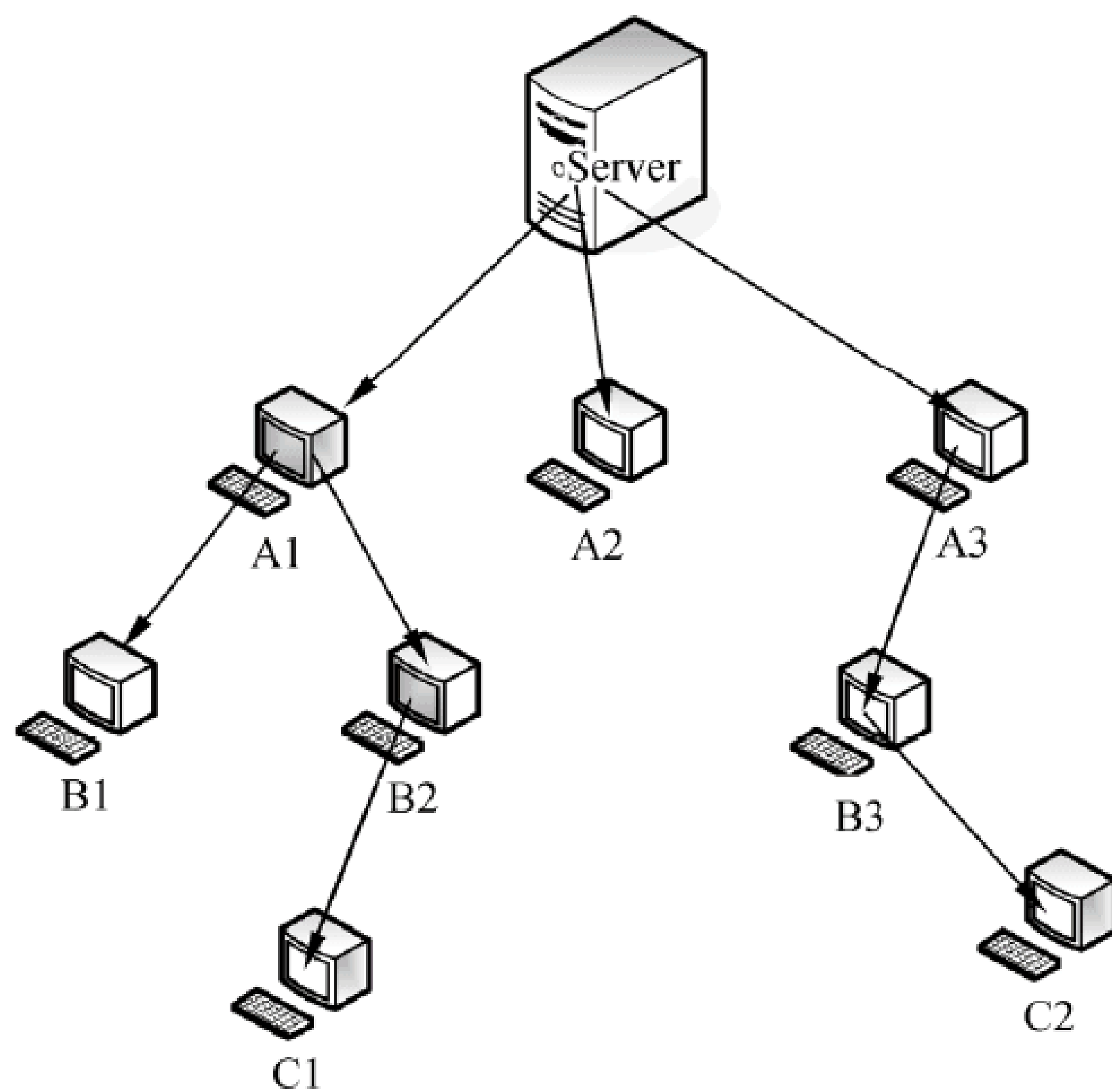


图 9.9　树状结构数据传输拓扑图

基于单组播树的数据分发方法研究的主要问题，是如何设计组播树构造方法和组播协议，以达到特定的性能指标要求。此外还包括如何增强系统的容错性，即如何减少结点离开或失效行为对其他结点的影响。

在树结构的分发策略中，由数据发布源（服务器）组织一个树结构的逻辑覆盖网络，数据由 S 开始依次向下传输，其数据分发的基本过程是。

- 新结点加入时从源服务器（即图 9.9 中的 Server 根结点）出发沿组播树进行搜索，直到发现某个非饱和结点为止。
- 新结点找到一个非饱和的结点时，将作为该结点的子结点加入到系统中。
- 当结点主动离开系统时，首先向其父结点进行注销并向其每个子结点发送一条重定向消息，以便让它们直接加入离开结点的父结点或从根结点重新开始加入搜索过程。
- 对于结点的失效：在所有父子结点之间维持心跳功能，如果在某段时间内收不到父结点的心跳信号，就认为父结点已经失效并重新向根结点发起加入搜索过程。

注意： 心跳，是用来判断对方（设备，进程或其他网元）是否正常运行，而定时发送简单的通信包。心跳机制一般用来监控集群中结点的状态。在结点间信息通信、故障判断、事件触发等方面有重要的作用。

以上的这个过程中，存在一个很重要的问题就是数据延迟，当父结点退出或失效时，将暂时中断子树上的结点数据传输。此时，组播树将迅速地重新维护这个棵树的结构，这个过程将耗费一定的时间，导致子孙结点服务被中断的时间较长，造成数据传输延迟。

2．多组播树的分发策略

多播树结构在数据的传输路径上引入了冗余，视频流不再由一个组播树来完成转发，而是先按设定的编码方式被分割成视频段，然后不同视频段由不同生成树完成转发。这样可以做到一个结点仅在一个组播树中作为转发结点，在其余组播树中作为叶结点。当一个

结点出现死机或网络出现故障时，只影响小部分结点。

基于多组播树的流媒体系统实现方案典型代表是微软研究院的 CoopNet 和 SplitStream，这两个方案都建立在多描述编码 MDC（Multiple Description coding）的基础上。多个组播树结构的最大好处在于它能提供良好的容错性，这对于拓扑结构呈动态变化的对等网络系统来说是非常重要的。

采用 MDC 编码的媒体流，这些 MDC 流在解码时不存在依赖关系，并分成多个层分别传输于多棵树上，每个结点从它所在的多棵树上获取数据，然后再将各层数据整合，还原成可以播放的媒体数据，如图 9.10 所示。

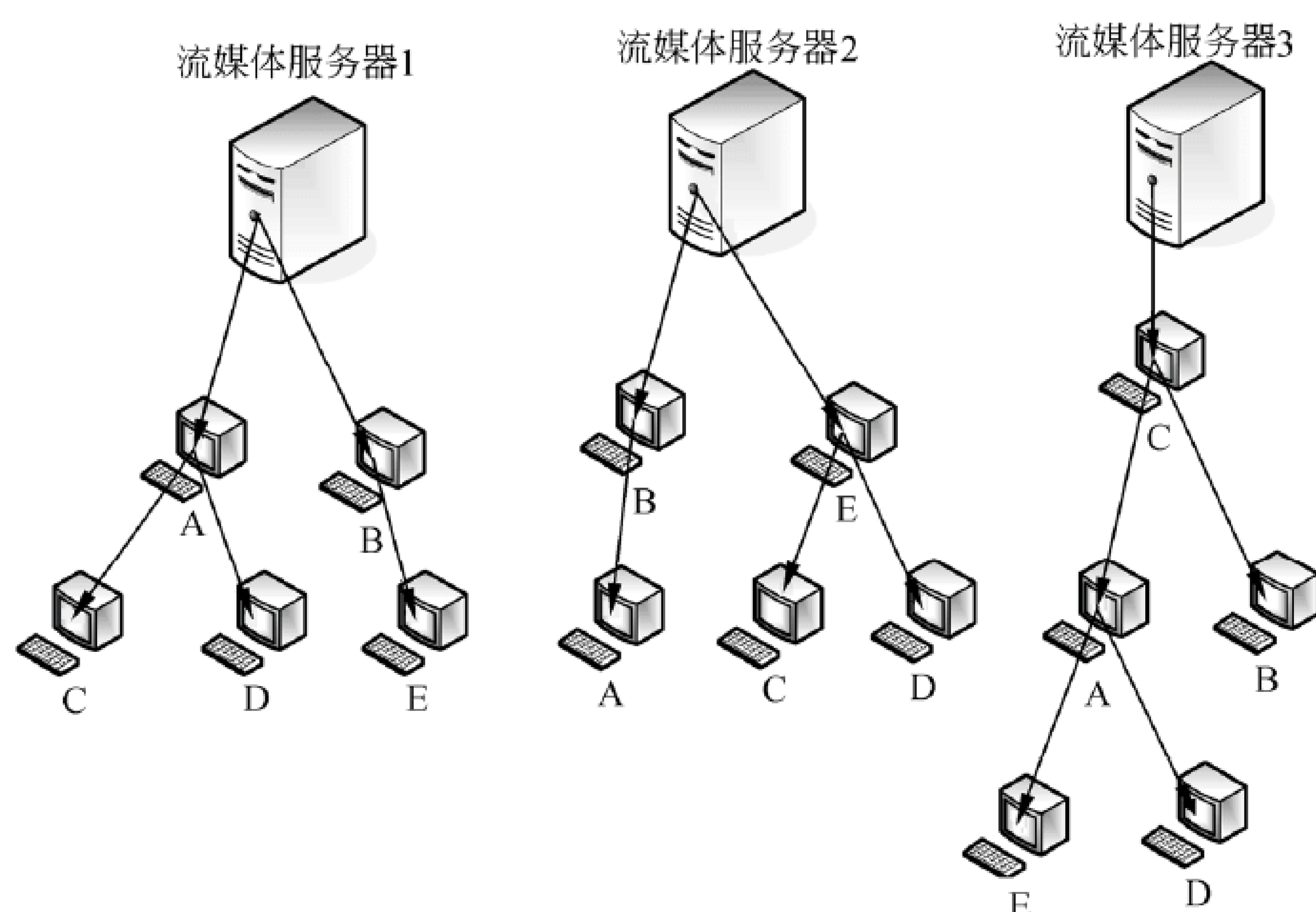


图 9.10　多树结构的数据传输拓扑图

在多组播树的系统中，对任意一个结点而言，其接收到的 MDC 子数据流的数量越多，则播放质量越好。当某棵组播树上的父结点离开或失效时，只会导致一条 MDC 子流的数据传输被中断，结点还可以从其他组播树上继续接收其他的 MDC 子流。在这种情形下除了画面质量会有所下降外，某些层的数据缺失不会造成媒体数据无法播放，只会影响其播放质量，而不会造成播放的中断或太长的延迟。

对于多棵组播树的组织和维护，主要由中心索引服务器集中承担，结点在加入或重新加入时均向中心索引服务器提交请求，以获取多个或单个父结点。此外，针对结点带宽资源异构的问题，有的文献上还提出了一种基于分层 MDC 编码的多棵组播树数据分发方案，以解决不同结点对不同播放质量等级的要求。

3．基于网状结构的分发策略

无论是单组播树拓扑还是多组播树的流媒体分发方案，即使采用了一些组播树维护方法，但在系统结点存在较高扰动的情况下，依然不能满足视频播放 QOS 的要求。对此，众多基于网状拓扑结构的流媒体分发方案涌现出来，较为典型的系统包括 DONET、AnySee 等。如图 9.11 所示为基于网状拓扑结构的流媒体分发。

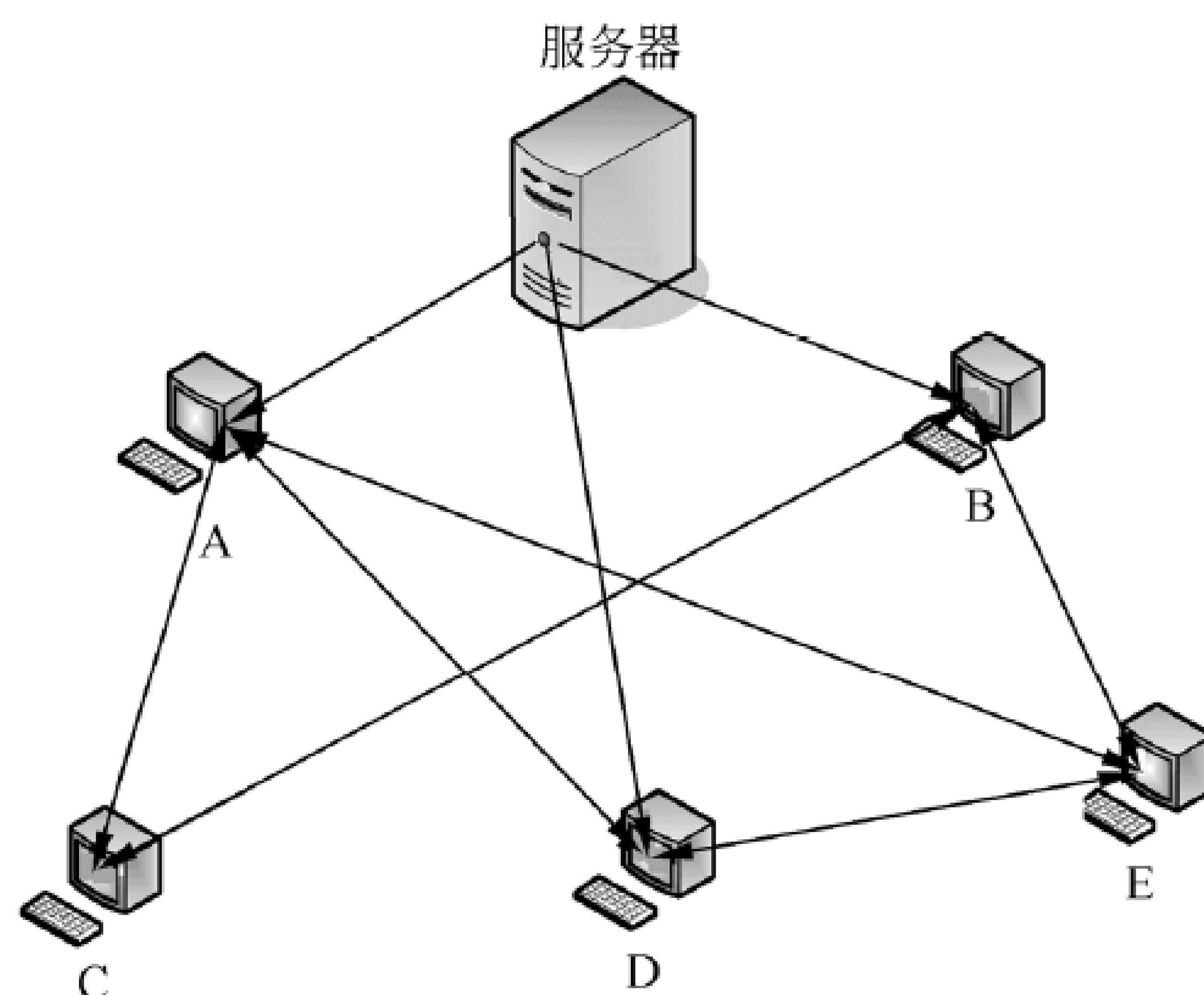


图 9.11　网状结构数据传输拓扑图

这些系统的主要思想均为利用 Gossip 协议来构造一个随机拓扑结构。在传输过程中，每个结点交换本地已缓存的数据位图 BM，结点根据播放进度和数据位图向邻居结点发起数据请求，从多个结点并行接收媒体内容数据，从而对每个结点来讲，形成一个多对多的数据传输模型。

网状模型可以把负载分散到每个结点上，适用于能力比较低的结点，比如 PDA、手机、ADSL 上网的 PC 机等。在这些情况下，单个结点不足以提供媒体播放所需的带宽 R，可以通过多个结点同时给结点提供数据，使其输出带宽之和大于 R，以支持正常媒体服务。但是，网状模型的流媒体系统通常都需要比较大的缓存，并且时间延迟比较大。

9.3.2　数据处理与编码技术

目前网络带宽仍是网络发展的主要问题，它一定程度上限制了包含大数据量的多媒体的传输，因此必须对多媒体数据进行预处理才能适合流式传输。预处理主要包括两方面，一是降低质量；二是对数据进行编码压缩。

在当前人们对媒体数据要求越来越高的情况下，降价媒体数据质量并不是一个好的方案，因而将数据的预处理方法就集中在对数据的编码压缩上。

1．媒体数据的编码要求

在流媒体系统中，对数据编码要一个明确的要求，主要体现在，要尽可能地提高视频数据的压缩效率，尽可能消除视频中的冗余信息，使用信源编码输出的数据量尽可能地少，以适应网络最低传输带宽的情况。同时，要提供灵活的视频质量分级来最大限度适应网络的动态特性，尽可能为用户提供好的视频质量。

在常用的编码技术中，对流媒体文件主要采用层编码的方式。采用层编码技术应用在 P2P 流媒体中，用于对不同能力的结点提供不同的流媒体服务。通过层编码把文件分成 0～n 层，每个结点都可以选择下载其中的 0～k 层（k≤n）。即便只有下载第 0 层也能够实时播放，但层数越高播放的品质越好。每个可以选择适合自己性能、带宽的层数，实现不同 QoS 的流媒体服务。

2．常用的编码方案

为了应对网络的抖动、提供稳定可靠的视频流，流媒体的传输普遍采用了数据编码技术。分层编码（layered coding，LC）和多重描述编码（multiple de2scription coding，MDC）是两种最基本的不可靠信道多路径传输的编码方式，在 P2P 流媒体系统中被广泛采用。近年来，网络编码（network coding，NC）技术和 P2P 流媒体传输相结合提高了网络的吞吐能力。

其应用方式是源点先把视频流分成小数据块，再对多个块进行编码后发出。接收结点只有在收到足够的线性无关数据包并解码之后才能获得原始数据。同时因为经过编码之后的每个块中都包含了多个原始块的信息，在一个视频段内不存在需要先后调度的问题，解决了分段传输的调度问题。但网络编码引入了编解码的开销，且需收集到一段中的足够多块才能解码，因此会带来额外的延迟。

9.3.3　媒体文件的定位技术

对于实时性要求相对较高的流媒体服务而言，媒体文件的定位至关重要，怎样快速地找到可以为自己提供数据的结点，怎样获取性能较好的结点，都是影响流媒体质量的关键。

注意： P2P 流媒体技术中，媒体文件的定位技术其实就是 P2P 网络中的结点搜索技术，关于 P2P 网络的搜索技术在本书相关章节中有过详细的讲解。这里只是针对流媒体结点搜索的一些特性进行针对性说明。

基于 P2P 的流媒体的文件定位方式显然取决于 P2P 网络的类型，目前 P2P 网络的文件定位方式主要有集中目录式、洪泛式、DHT 等几类。

1．集中目录的定位方式

集中目录式是指结点对于文件的请求都会发送到一个请求给中央目录服务器，目录服务器存储了整个网络的索引信息，目录服务器收到请求后返回拥有该文件的结点信息（如 IP 地址等）。请求结点收到回复后选择其中结点单独建立连接。

这种方法的优点是实现简单，目前国内非常成功的流媒体软件 PPLive 和 PPStream 都采用了这种方式。它的缺点是中央服务器的瘫痪容易导致整个网络的崩溃，可靠性和安全性较低，而且随着网络规模的扩大，对中央索引服务器进行维护和更新的费用将急剧增加，所需成本过高。

2．洪泛定位

洪泛式定位方法，是指结点通过广播加 TTL（Time to Live）的形式请求文件，在 P2P 网络的典型应用是 Gnutella。GnuStream 就是利用 Gnutella 网络的搜索方式来实现文件定位的，然而洪泛查找的效率并不高，而且还会造成大量的冗余信息加重网络的拥塞，在些无结构 P2P 网络中改进了洪泛搜索的方法，采用概率洪泛（probabilistic flooding），通过概率分析选取分支进行洪泛，极大地提高了文件定位的效率。

3．DHT定位方法

DHT 通过 Hash 算法实现，每个文件经过 Hash 运算后得到一个唯一的 ID，每个结点也对应一个 ID，文件存储到与其标识符邻近的结点中，在不需要服务器的情况下，每个结点负责一个小范围的路由，并负责存储一小部分数据，从而实现整个 DHT 网络的寻址和存储。

查找文件时，首先对文件名进行 Hash 运算得到该文件的 ID，通过结点的路由表信息查找存放文件的结点。由于基于 DHT 的 P2P 网络能够自适应结点的动态加入/退出，有着良好的可扩展性和鲁棒性。结点 ID 的分配具有均匀性和自组织能力。

目前很多流媒体模型都建立在诸如 Chord、CAN、Pastry 等 DHT 网络之上，所以 DHT 方法的文件定位方式在 P2P 流媒体中的应用非常广泛。Promise、PROP、MSMC 等模型都属于此类。DHT 的缺点是仅支持精确关键词匹配查询，无法支持内容/语义等复杂查询。

4．其他定位方式

除了以上的几种 P2P 网络常用的结点搜索方法外，对流媒体网络中还有其他的一些媒体文件的定位方法。在一般树形结构流媒体网络中，结点搜索查找媒体文件的时候，会自组织进入组播树中，通过一种分等级的覆盖网结构找到自己合适的父结点，具体实施方法因不同树的结构而异。PeerCast 的结点是自上而下加入的，先加入结点位于树的高层，ZIGZAG 则是自下而上的，所有结点加入到最底层。

以 ZIGZAG 为例，新加入的结点会向源服务器发送请求，服务器将请求沿着已有的组播树层层下传，直到找到最底层的未满的簇。由于 ZIGZAG 本身的心跳机制互通控制信息，簇首很容易找到未满的底层簇，所以加入结点也能很快地加入组播树获得媒体文件。

9.3.4　媒体同步与拓扑均衡性

由于网络时延，导致媒体流在传输过程中失去同步关系，传输的时延不可预期，媒体同步机制可以确实地恢复媒体流的同步。

1．媒体同步机制

同步机制的目的就是保证接收端以正确的时间收到的媒体数据。媒体同步机制实际上就是在媒体内或者媒体间说明。

对于连续媒体，应用最为广泛的说明方法说明或时间戳。时间戳法是在每个媒体的数据流单元中加进统一的间码，具有相同时间戳的信息单元将同时予以表现。在发送时，将按时间顺序分成单元，在同一个时间轴上，给每个单元都打上一个同一时标的各个媒体单元具有相同的时间戳。在各个媒体到达终端相同时间戳的媒体单元同时进行表现，这样就得到了媒体之间的同步。

2．拓扑均衡性问题

在 P2P 网络中每个结点的能力都是不同的，结点的能力包含运算速度、上下行带宽、在线是否稳定等各个方面。对于流媒体应用系统这种对稳定性和实时性要求较高的系统，

与不同性能的结点怎样采取不同的策略使得它对系统的影响减到最低，是要解决的首要问题。目前一般有两种策略。

- 策略一：在同等服务质量下，能力越强的结点担负更大的责任。
- 策略二：同等作用下，为不同能力的结点提供不同品质的服务。

策略一的关键在于结点的选择，根据不同的场合会有不同的选择策略。

- 如果希望流媒体的延时小，可以选择物理邻近的结点承担更多的转发任务。
- 如果希望得到高质量的流媒体服务，高带宽和 CPU 能力强的结点应当被选择承担更大的责任。
- 如果希望获得稳定的流媒体服务，在线时间更长的结点则是最佳选择。

在一般情况下，碰到更多的是以上各种情况的折中。事实上，大多数 P2P 流媒体类型就是这样做的。在树形结构中，如 ZIGZAG，每个簇的簇首和连接结点是通过对其不同能力的比较产生的，簇首会晋升到树的上层，成为层际转发的枢纽。而且还要承担管理维护算法，所以选择高带宽和高运算速率是必须的。而连接结点是本层内的转发结点，与本层结点的邻近成为了衡量它能力的最主要标准。

而在 Promise 为代表的多对一网型结构中，结点都会通过查找获得目标结点列表，然后选择若干结点作为父结点，于是能力更强的结点无疑会有更多的机会被选中，自然在 P2P 网络中承担更多的转发任务。通过不同的 QoS 服务，也是解决结点不均衡性的发法。

9.3.5　媒体数据的交换技术

P2P 流媒体系统在建立了覆盖网之后，需要考虑在覆盖网之上选择路径来传输数据。数据的交换技术存在着“推”和“拉”两种机制。

所谓“推”，就是结点主动向另一个结点发送数据，这需要结点之间有一种父与子的关系，父结点依据这种关系主动发送数据给子结点。所谓拉，就是结点首先向另一个结点发出请求，另一结点再根据请求发送数据，这不需要结点之间任何层次性的关系，但是结点需要预先知道对方的含有的数据。一般组播树是典型的推机制，而纯的 P2P 则采用拉的机制。

1．推机制

推的机制需要结点之间具有父与子的关系，父结点负责为子结点发送数据。因而父结点失败时就必须为子结点选择新的父结点。在这种关系调整的时候就可能导致子结点丢失部分数据，因而树结构的机制很难保证质量。

此外树结构一般从一个结点请求数据，结点与结点之间带宽一般比较低因而只适合带宽要求低的应用。后来又提出了采用多描述编码采用多棵树的机制，通过将单一码流分割为多个码流，对每个码流建立一棵传输树。多树的机制对质量有所改善，但是其算法的复杂度和维护的开销大大增加。

2．拉机制

拉的机制是 P2P 系统中常用的方法，它以存储转发为基础，结点首先获得对方结点所拥有的数据状态信息，然后向其发起请求。拉的机制无须维护树的结构，但是结点之间需

要不停地交换缓冲区中数据的状态信息而产生一定的负载。

拉的机制可以使得结点可以从任何相邻的结点在任何时间获取自身需要的数据，因而算法上实现简单而且可以提高自己的下载带宽。

推和拉的机制各有特点。推的机制无须发送数据的状态信息，无须存储转发因而负载较低且延迟较低。当然推的机制需要维护复杂的树结构，而且对动态变化的应对能力比较弱，结点只能从单个结点获取数据，导致采用推的机制质量更差，结点获取数据的能力更差，实现上更为复杂。

拉的机制采用存储转发的思想，需要数据状态的转发，需要在每个结点进行必要的数据缓冲，因而拉的机制导致较长的时延和较多的负载。当然，拉的机制可以随意同时从多个结点获取数据，结点之间完全对等，其实现也更为简单，更容易处理结点的高动态性，也能获得更高的下载带宽达到更好的质量。

9.3.6　媒体数据的缓存技术

因为 Internet 是以数据包传输为基础进行断续的异步传输，原始数据在传输要被分解成许多包，由于动态变化的网络，各个包选择的路由和到达客户端的延迟可能就不相同。为此，要尽量消除弥补延迟和抖动的影响，可以使用缓存技术，这样能保证数据包的连续输出和顺序正确，从而不会使播放出现停顿。

P2P 流媒体系统的一个基本需求是数据冗余 P2P 流媒体系统会将视频分为较小的数据块，结点将试图预取和缓存一些数据块以应对网络的动态变化，这需要两个关键策略，即缓存替换策略和缓存——中继策略。

1．替换策略

有限的缓存空间要求系统必须具有缓存替换策略，相关问题包括替换时机问题，即确定在何时启动替换过程；另一个就是替换对象问题，即确定丢弃哪些数据块。不恰当的替换策略导致不必要的服务器请求。一些通用的缓存替代策略如下。

- 最少最近使用（LRU）：替换最长时间没有被请求的数据块；
- 最少频繁使用（LFU）：替换被请求次数最少的数据块；
- 数据的大小：替换缓存中最大的数据块；
- Hyper2G：这个策略是 LRU、LFU 和大小策略的综合；
- GreedyDual：每个缓存中的数据块被指定一个效用值，替换效用值最小的数据块。

大多数替换策略可以归类为使用一个效用函数。传统的流媒体系统普遍采用的缓存替换方案是 LRU，但 LRU 可能导致流行的媒体块被大量缓存，而不流行的媒体块最后从 P2P 网络中消失。在 P2P 流媒体中，通常采用给流行数据块以较高优先级的效用函数，这有可能带来视频不完整的问题。

2．缓存——中继策略

早期的 P2P 流媒体系统大多是直播系统，直播系统由于播放时序的同步性以及网络的传输延迟，所有结点只需也只能缓存数据源服务器播放点之后的数据块。相比于整个流媒体文件的长度，这个缓存目标相当小，因而可采用缓存－中继的策略。

缓存——中继的策略的主要思想是，接收者以滑动窗口的形式缓存最近播放过的内容，然后将这些内容提供给系统中那些在一定播放延迟之内的请求结点。

- 异构性，它是一个应用层的解决方案，对底层网络没有特殊要求，无须特定的代理。结点播放并缓存媒体内容的少量数据，通过从彼此的缓存中获取流数据形成了协作式覆盖网络。
- 可扩展性，流覆盖网络具有随着客户数量扩展的能力，大大降低了对服务器性能和带宽的要求。或者说，相比于传统的流媒体系统，该策略能够以同等性能源服务器支持数量级更高的结点群。
- 流质量的适应性，每个客户可以根据不同的 QoS 需求选择流源。

缓存——中继方法也存在两个问题：首先，一个结点离开系统时，任何从该结点接收数据的结点将被中断服务。第二，断连的结点被看作新结点，增加了服务器和网络的负载。

因此，“缓存——中继”策略从仅缓存已播放数据块被扩展为可使用额外带宽预取部分未播放数据的“预取——缓存”策略。

采用“预取——缓存”策略，结点在其播放点之前预取和缓存流媒体部分数据，以确保结点能应对网络抖动和服务结点离开造成的影响，使媒体流得以平滑播放。同时，预取的数据块也可服务于网络中的其他结点，降低内容分发的压力。虽然预取需要额外的带宽和缓存空间，但由于 Internet 中结点网络带宽和存储能力不断增加，“预取——缓存”策略实际上提供了一种在播放质量和播放成本间进行权衡的有效方法。

9.3.7　P2P 流媒体系统的重要机制

在 P2P 流媒体系统中，为了进一步完善相关功能引入了很多机制，这些机制对系统的稳定性、安全性、健壮性有着重要的作用、这些重要的机制主机有容错机制、安全机制、激励机制等。

1．容错机制

P2P 流媒体系统拓扑具有很强的动态性，结点的加入和离开是不可预期的，传输的连接很可能会突然失效，此外一些链路也可能因为拥塞而出现丢包的现象，这些问题也会影响到流媒体系统的稳定运行。增强 P2P 系统的容错性是非常重要的。

当发生结点失效时，目前不同的 P2P 流媒体系统都会有相应的应对措施。对于采用树结构的网络，通常要重新运行选择算法寻找新的父结点。经过算法上的优化（如 ZIGZAG），可以达到较快的重连速度。网型结构的网络由于是多对一传输，在结点列表中会有一些备用的结点，失效时马上连接备用结点，可以更快捷地继续提供服务。

在混合型的网络中，Proxy 的存在会大大提高容错性。作为代理结点，它具有较高的稳定性。用户在重新选择父结点时可以暂时从代理结点的缓存中获得所需数据。当然无论采取何种应对措施，如何在第一时间检测到失效结点，如何在连接切换时不影响流媒体的连续性，是问题的核心，也是未来继续研究改进的重点。

对于数据包丢失的应对也是衡量系统容错性的环节之一。这可以通过数据编码技术来解决，比如前向错误编码（FEC）和多描述性编码（MDC）。Promise 模型中采用了 FEC 算法，它通过给压缩后的媒体数据编码流加入冗余信息的方法，使得在一定范围内的丢包

率下，用户仍然可以还原出原始数据。

例如在 Promise 中采用 FEC（a）算法，用户可以容忍最高（a-1）%的丢包率。即当 a=1.25 时，丢包率上限是 25%。SplitStream、CoopNet、MSMC 等模型则采用了 MDC 算法，它是对同一媒体流内容采用多种方式进行描述，每一种描述都可以在一定解码质量下单独解码。多种描述方式结合起来则可以增强解码质量，这也意味着即使一定程度的丢失数据包，用户还是可以以降低流媒体品质的方式解码数据，不会影响流媒体服务的连续性。

2．激励机制

P2P 流媒体系统是一个基于结点的共享系统，大部分的资源和服务都分散在网络中的各个结点上，而结点的加入和退出在时间和空间上都是高度自由的。要保证整个系统的健壮稳定运行，需要引入激励机制，以最大限度的鼓励结点之间的数据共享。

目前的各种激励机制主要有以下 3 种。

- ❑ 基于微支付机制：基于微支付机制用虚拟货币作为 P2P 网络中服务或资源交易的媒介，体现了对结点贡献的反馈，从而使得结点有积极性去参与合作。
- ❑ 基于直接互惠的机制：基本思想是 P2P 网络中的服务提供结点，在为其他结点提供服务后能得到某种直接优惠。
- ❑ 基于信誉机制：主要是在 P2P 网络中引入了一个等级的概念，即每一个结点根据自己在网络的历史行为情况，获得由网络中与它邻近的其他结点所评价得出的信誉值。在以后的服务或资源交易中，其他结点均根据请求结点的信誉值给予对应等级的回应。

利用这种激励机制，可以鼓励用户尽量在线，使每一个结点尽可能参与到流媒体网络的共享中。

3．安全机制

网络安全是 P2P 流媒体系统的基本要求，通过安全领域的防火墙、身份识别认证、授权、数据完整性、保密性对 P2P 信息进行安全控制。数字版权管理（DRM）可以有效保护知识产权，通过 DRM 技术，内容提供商可以方便地对各种音乐、图像等媒体文件进行加密保护，使受保护的多媒体文件不会被用户非法拷贝和复制。在 P2P 流媒体系统内，可采用用户分级授权的办法，阻止非法访问。

以上只是对 P2P 流媒体中一些关键技术的简要介绍，一些重要的技术和思想，读者还需要在此基础上进行深入学习才能掌握。总地来说，基于 P2P 的流媒体技术代表着未来多媒体数据在网络传输的发展方向，基于此技术的软件产品诸如 PPLive、PPStream、QQLive 已得到广泛的应用，并显示出越来越好的发展前景。在这 P2P 流媒体这一技术体系中，已经有了很多经典、成熟的技术，并且得到了实践的检验，也有很多需要解决的问题，这里有很深的研究空间、有巨大的潜力可挖。

9.4　P2P 流媒体的应用及典型的应用系统

在 P2P 流媒体的世界里，人人都是源头、人人都是受众的优质客户，它以其优异的性

能和出色的表现，正在互联网的流媒体发布中扮演越来越重要的角色。基于 P2P 流媒体技术的各类软件、系统平台也如雨后春笋般在 Internet 的几乎每个角落生根发芽，尽管有些特征还未完全显现，但不可否认的是，P2P 流媒体技术所带来的是一场新媒体的革命，基于它的应用技术也正在促生着一个全新的互联网。本节就重点讲一下 P2P 流媒体的应用技术和它典型的应用平台。

9.4.1　P2P 流媒体技术主要应用方面

网络的迅猛发展和普及为 P2P 流媒体业务发展提供了强大市场动力，P2P 流媒体技术的应用则为网络信息交流带来了革命性变化。目前常见的 P2P 流媒体的应用主要有下面几种。

1．视频点播（VOD）

视频点播技术是最常见、最流行的 P2P 流媒体应用类型。视频点播技术简称 VOD，也称为交互式电视点播系统。VOD 出现的最初动力是人们对广播电视的不满，在现行的电视节目中，收看者完全是被动的。节目提供者放什么节目，观众就只能看什么节目，节目时间也是固定不变的。尽管电视台可以提供很多的节目，但要想真正完整地收看到一个自己满意的节目，对于许多人来讲也是不太容易做到的，因为在快节奏的现代社会中，许多人不可能为了看某一个电视节目而预先安排自己的时间。被迫习惯了这种被动收看方式的人们，对于有朝一日能够按照自己的需要自由地点播，充满了美好而迫切的憧憬。

P2P 技术与流媒体点播业务的结合，促进了 VOD 技术的进一步发展，它突破了传统点播系统在带宽、负载能力、处理能力方面的限制，真正实现了观看的人越多、视频节目越流畅的目的，在此背景下，基于 P2P 技术的各类视频点播系统开始出现在互联网上，当前已有越来越多的网站提供在线视频点播功能。

2．视频广播（直播）

视频广播，也叫视频直播，这是一种实时的、现场的视频数据的发布形式，可以看作是视频点播的扩展，它把节目源组织成频道，以广播的方式提供。在这种方式下，通过一个单一的源，可以把直播视频节目广播到网络中的所有结点上，在体育比赛和重大活动直播、网络电台、影视节目轮播等方面有重要的应用。

3．交互式网络电视（IPTV）

IPTV 是 Internet Protocol Television 的缩写，是一种系统的总称。在这一系统中，电视和视频信号使用因特网协议上的宽带连接分配给订户，这经常是与订户的因特网连接并行的，由宽带运营者使用相同的基础设施提供，但在专用带宽分配之上。IPTV 的网络结构如图 9.12 所示。

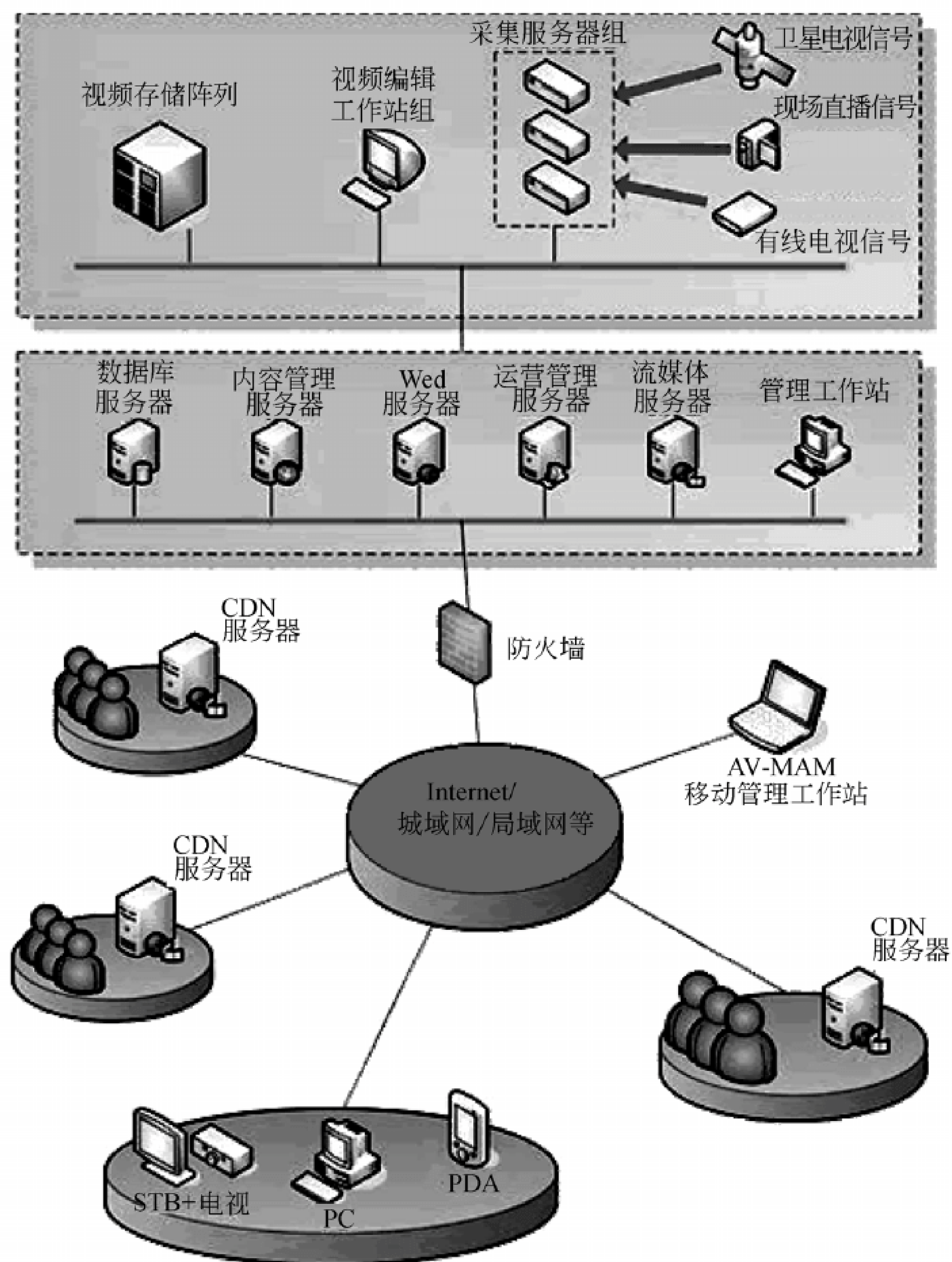


图 9.12　IPTV 的网络结构示意图

IPTV 利用流媒体技术通过宽带网络传输数字电视信号给用户，这种应用有效地将电视、电信和计算机 3 个领域结合在一起。在基于 P2P 的平台上，最大限度的节约了带宽，保证了视频节目的传输质量，P2P 与 IPTV 技术的结合具有很好的发展前景。

4．远程教学

远程教学目前应用也比较广泛，而且具有很好的市场应用前景。远程教学可以看作是前面多种应用类型的综合，在远程教学中，可以采用多种模式，甚至混合的方式实现。远程教学以应用对象明确、内容丰富实用、运营模式成熟，成为目前商业上较为成功的流媒体应用。

5．交互游戏

传统的网络游戏都是用 C/S 模式来架构的，这样实现起来比较简单。但是服务器端的资源是有限的，当游戏用户过多时，就会给系统服务器、带宽带来较大的压力，所以对于

一些不重要的数据，也可以用 P2P 的方式来发送。比如同一画面上玩家的位置信息，它只需要跟与它在同一画面上的其他玩家同步位置，服务器再做简单验证就可以了。这种通过流媒体的方式传递游戏场景的交互游戏近年来得到了迅速的发展，如跑跑卡丁车、一些网络休闲游戏都用到了 P2P 的流媒体技术。

其他流媒体系统的一些新的应用和服务，例如虚拟现实漫游、无线流媒体、个人数字助理（PDA）等也在迅速地变革和发展。

随着 P2P 技术的不断成熟和发展，基于 P2P 流媒体的系统平台、P2P 流媒体应用软件等，如雨后春笋般出现。这些应用不仅在商业上取得巨大成功，也给网民带来了实实在在的好处，使他们能在互联网上可以轻松的看到高质量、高清晰的视频画面。

流媒体由于加入了 P2P 技术而得到蓬勃发展，随着网络电视 IPTV、无线流媒体、数字家庭等未来流媒体的应用，相信 P2P 流媒体还将会有一个更广阔的前景。

9.4.2　P2P 流媒体的典型应用系统

在 P2P 流媒体发展及走向应用的过程中，出现了大批经典的应用系统。除最早的一批以 PPLIVE、PPstream 为代表的 P2P 流媒体之外，大批的 P2P 网站、P2P 网络电视、P2P 视频点播系统等相继诞生，QQLive、UUsee 、PPMate、SopeCast 等均借此契机声名大噪。有数据显示，这一时段产生的 P2P 流媒体网络公司达 200 家之多。本节就对这些典型的 P2P 流媒体应用系统进行简要的介绍。

1．PPLive网络电视

PPLive 是一款用于互联网上大规模视频直播的免费共享软件，有别于其他同类软件。PPLive 内核采用了独特的 ALM 多播和内聚算法技术，有效地降低了视频传输对运营商主干网的冲击，减少了出口带宽流量，并能够实现用户越多播放越流畅的特性，有效解决了当前网络视频点播服务的带宽和负载有限问题，使得整体服务质量大大提高。此软件在百度软件风云榜的排名居高不下。TOP50 的上榜天数长达近一年时间。如图 9.13 所示为 PPLive 系统运行界面截图。



图 9.13　PPLive 系统界面截图

与传统的网络电视相比，PPLive 的优势在于其巨大的技术优势。在网民心目中，以往想在网络上看电视，只能登录提供网络电视服务的网站，然后通过 Real player 等播放器来播放在线视频，用户一多，画面和声音都会非常“卡”。而 PPLive 采用的是比较前沿的 P2P 技术，根据 P2P 的原理，用户越多，速度反而越快，彻底改变了用户量和网络带宽之间的老大难问题。同时，在同类的网络电视软件中 PPLive 自主开发的技术也走在了潮流的最前面。PPLive 有效的解决了内网穿透问题，开发出目前最领先的“穿透内网自动打开 UPnP 功能”技术，并解除 WinXP 对 TCP 的连接数的限制。还有效地使得 PPLive 能够方便地穿透防火墙。以上种种都为局域及各种内网用户提供了最大程度上的便利。

PPLive 节目来源多源化，有的是网友自己制作的，有的是合作伙伴内容提供商提供的，有的是广电和电信运营商提供的，内容积极健康，各种用户喜爱的节目在 PPLive 的平台上都可以找到。良好的节目来源也决定了 PPLive 并不会涉及到任何版权纠纷。

2．PPStream

PPS 网络视频（WWW.PPS.TV）创办于 2006 年 1 月，是全球最大的网络视频软件服务运营商。

2005 年 6 月，PPS 在雷量、张洪禹两人的研发下，正式推出，立刻获得了惊人的肯定。2005 年 9 月徐伟峰加入，正式进入了的商业运营并着手成立公司，三位联合创始人从此成为互联网影音视频的领导者。PPS 在创立的第一天就将自己定位成未来的互联网电视平台，并愿为此目标不懈地努力奋斗。

PPS 一直致力于倾听、挖掘与满足中国网民的需求，秉承“用户体验至上”的理念。除 PPS 网络视频播放器外，还提供影视百科、PPS 看看、娱乐、社区、图库、影视调查等多样化的产品及服务，率先创造了以影视百科为代表的社区，将无数网民头脑中所有的影视需求融入了 PPS。PPS 已经成为了人们进行互联网影视的代名词。如图 9.14 所示的是 PPStream 的界面截图。



图 9.14　PPStream 系统界面截图

PPS 网络视频目前已拥有 30000 套频道节目，总安装量已达 3.5 亿，日活跃用户达到 1100-1200 万人，月活跃用户超过 1 亿，月收视时长突破 600 亿分钟，社区注册用户 600 万，并稳居网络视频第一。

3. QQLive

腾讯公司成立于 1998 年 11 月，是目前中国最大的互联网综合服务提供商之一，也是中国服务用户最多的互联网企业之一。

QQLive 是由腾讯公司自主研发的 P2P 流媒体互动传播平台。QQLive 能够为互联网用户提供稳定和流畅的音视频直播节目。与传统的流媒体相比，QQLive 采用了 P2P-Streaming 技术，具有用户越多播放越稳定，支持百万级用户同时在线的大规模访问等特点。QQLive 客户端可以应用于网页，桌面程序等各种环境，并提供丰富的视频节目及实时互动功能，能够满足不同类型的用户需求。如图 9.15 所示为 QQLive 系统界面。



图 9.15　QQLive 视频播放系统

截止 2008 年 12 月 31 日，腾讯即时通信工具 QQ 的注册账户总数已经达到 8.919 亿，活跃账户数达到 3.766 亿，最高同时在线账户数达到 4970 万。这些数据足以说明 QQLive 在网络视频方面有稳定的市场也有足够强的代表性。

QQLive 借助 QQ 平台，在互联网中非常流行，有广大的用户群，同时，QQLive 提供了丰富的视频资源，且更新速度，视频质量也相对较高。还有一点就是，QQLive 的视频特征全面，即有直播节目也有点播节目，而且节目之间有明确、清晰的分类，涉及的视频领域包括在线电视、影视、剧集、综艺、文化、记录片及学习视频等，在整个基于 P2P 的网络视频中既有自身的特色也有普遍的代表性。

4. PPMate网络电视

PPMate 网络电视是一款用于互联网上大规模视频直播的软件，PPMate 是一款免费绿色的视频内容聚合客户端。它不同于其他 P2P 流媒体系统的是，PPMate 提供的内容均系

程序在网络上自动收集，版权归内容所有者。PPMate 仅为网友提供视频交流平台，不存储任何视频内容。PPMate 的节目列表由 PPMate 爬虫程序自动采集网页生成。PPMate 不保证所提供的文字描述和实际内容相符。

PPMate 在开发及推出时充分考虑用户的感受，不包括任何插件、木马等程序，软件中已经聚合数百个电视台，包括了海外的 HBO、好莱坞、ESPN 亚洲，法国时尚……；港台地区的凤凰卫视、电影台、翡翠台、龙祥……以及国内的中央和地方各个上星节目。囊括了几乎所有知名电视台，全心提供直播比赛、震撼科幻惊悚电影大片、最新新闻以及国内最知名的导演和影视明星的专栏节目，灵活方便地满足用户选择。

5. UUSee 网络电视

UUSee 网络电视 2008，是悠视网打造的一款全新网络电视收看软件，用户使用这款软件可以免费收看 500 多路新颖频道，共计 1000 多个精彩节目。其官方网站为 http://www.uusee.com，UUSee 在实际应用中，主要有以下几个特点。

- 多模式播放：UUSee 网络电视 2008 提供精简/标准/全屏/双倍 4 种播放模式，窗口大小可以任意调节，置顶播放方式，避免无关操作影响观看。
- 高清晰视频：UUSee 网络电视 2008 基于 H.264 自主研发的编解码技术，无论是精彩激烈的运动场面，还是绚丽夺目的影视特技画面，都能呈现清晰流畅的播放效果。
- 酷热节目内容：UUSee 网络电视 2008 拥有丰富的影视剧版权资料库，向用户提供各种类型的最新、最酷、最热的节目内容，满足不同用户的需求。
- 快速录制：UUSee 网络电视 2008 为用户提供人性化快速录制功能，用户根据需要，既可以随时录制当前播放的节目，也可以定时录制自己喜爱的节目，省事省心。
- 准确的节目预告：UUSee 网络电视 2008 提供准确的电子节目单，显示所有频道，并可随时打开关闭，频道悬停显示当前节目及节目预告。
- 强大的视频流搜索：UUSee 网络电视 2008 集成目前最先进的视频搜索技术，搜索内容可精确到帧，精确定位海量视频内容，想看什么就看什么。
- 个性化频道管理：UUSee 网络电视 2008 提供的个性化频道管理功能，既能自动记录用户最喜爱的 10 个频道，又可以收藏自己最喜爱的频道，更可以创建自己的个性频道，随时收看，不必每次麻烦地寻找。

以上就是对几个主流的 P2P 流媒体系统的简要介绍，还有其他很多的 P2P 网络电视系统，如 SopCast、沸点网络电视、TVUPlayer、CCTV 网络电视、ETshow 网络电视、JooTV 聚视影视、Tvkoo 网络电视、搜狐电视机等，它们都是互联网上较活跃的 P2P 流媒体视频发布系统，有兴趣的读者可自行查阅相关资料以了解它们特点及性能。

9.5　本 章 小 结

本章从整体上介绍了基于 P2P 的流媒体技术，首先详细介绍了流媒体有关的知识，接着从 P2P 技术与流媒体结全的角度分析了 P2P 技术应用到流媒体中的一些特性和优势，并介绍了当前对 P2P 流媒体技术的研究情况。紧接着详细分析了 P2P 流媒体所涉及的关键技

术，最后介绍了 P2P 流媒体技术的应用情况及典型的流媒体应用平台。

通过本章的学习，读者应该掌握流媒体的基本知识，理解 P2P 技术应用到流媒体领域的原理和特点，重点掌握 P2P 流媒体所涉及的关键技术。

随着通信、计算机、多媒体技术的发展，以及安防、金融、教育等行业日益增长的客户需求，P2P 流媒体视频正在互联网生活的方方面面扮演着越来越重要的角色，学习 P2P 流媒体的相关知识有重要的意义。

第 3 篇 实战开发篇

第 10 章　基于 Java 的 P2P 开发平台搭建全攻略

从本章开始，就正式进入了 P2P 技术的实践开发部分。俗话说，“工欲善其事，必先利其器”，在正式讲解 P2P 的应用案例开发之前，有必要讲解一下 P2P 应用开发的工具和平台。从应用的角度来说，一个好的开发工具和平台，会让整个开发工程达到事半功倍的效果。所以本章就重点讲解一下进行 P2P 开发所需的工具及开发平台的搭建。

本章的重要知识如下。

- Java 编程语言：了解 Java 语言的基本特点，了解 Java 语言在 P2P 应用开发中的作用和地位。
- Windows 下安装 JDK 的方法：掌握获取并安装 JDK 的、配置环境变量、编译并运行 Java 源程序的基本方法。
- Eclipse 的安装与使用：学会如何获取和安装 Eclipse，会安装 Eclipse 插件，重点掌握 Eclipse 的使用方法。
- Linux 下搭建 Java 平台：掌握 Linux 系统下如何安装 JDK 和 Eclipse，了解如何在 Linux 系统下进行 Java 项目开发。

10.1　进行 P2P 应用开发的编程语言和平台

P2P 作为一种方法和原理性的技术，任何一种编程语言都可以开发 P2P 的相关应用，Java、C/C++、Python 等是最常用的开发语言，这些编程语言在 BT 系统、eMule 系统、流媒体平台、即时通信等方面都实现过相应的原型系统。本文中所有的实例都是用 Java 语言实现的，所以，本章所讲的主要内容就是如何构建一个基于 Java 的 P2P 应用开发平台。

10.1.1　Java 语言简介

Java 语言诞生于 1991 年，起初被称为 OAK 语言，是 Sun 公司为一些消费性电子产品而设计的一个通用环境。他们最初的目的只是为了开发一种独立于平台的软件技术，在 1995 年的时候，Sun 公司正式以 Java 这个名字推出该语言，从此，Java 成为编程界一颗耀眼的明星。

1．Java的定义

笼统地说，Java 是一种计算机编程语言，从特性上讲，它是一种简单的、面向对象的、分布式的、解释的、健壮安全的、结构中立的、可移植的、性能很优异的多线程的、动态的语言。

注意：学习本章及以后的知识，建议读者适当地学习 Java 编程语言的相关知识。

2. Java的特点

Java 语言作为一种广泛使用的计算机编程语言，本身具备了与其他编程语言相比很多的优异特点，要学好 Java 有必要了解这些特点。

（1）平台无关性：平台无关性是指 Java 能运行于不同的平台。Java 引进虚拟机原理，并运行于虚拟机，实现不同平台的 Java 接口之间。使用 Java 编写的程序能在世界范围内共享。Java 的数据类型与机器无关，Java 虚拟机（Java Virtual Machine）是建立在硬件和操作系统之上，实现 Java 二进制代码的解释执行功能，提供于不同平台的接口。

（2）安全性：Java 的编程类似 C++，学习过 C++的读者将很快掌握 Java 的精髓。Java 舍弃了 C++的指针对存储器地址的直接操作，程序运行时，内存由操作系统分配，这样可以避免病毒通过指针侵入系统。Java 对程序提供了安全管理器，防止程序的非法访问。

（3）面向对象：Java 吸取了 C++面向对象的概念，将数据封装于类中，利用类的优点，实现了程序的简洁性和便于维护性。类的封装性、继承性等有关对象的特性，使程序代码只需一次编译，然后通过上述特性反复利用。

Java 提供了众多的一般对象的类，通过继承即可使用父类的方法。在 Java 中，类的继承关系是单一的非多重的，一个子类只有一个父类，子类的父类也只能有一个父类。Java 提供的 Object 类及其子类的继承关系如同一棵倒立的树形，根类为 Object 类，Object 类功能强大，经常会使用到它及其他派生的子类。

（4）分布式：Java 建立在扩展 TCP/IP 网络平台上。库函数提供了用 HTTP 和 FTP 协议传送和接受信息的方法。这使得程序员使用网络上的文件和使用本机文件一样容易。

（5）健壮性：Java 致力于检查程序在编译和运行时的错误。类型检查帮助检查出许多开发早期出现的错误。Java 自己操纵内存减少了内存出错的可能性。Java 还实现了真数组，避免了覆盖数据的可能。这些功能特征大大提高了开发 Java 应用程序的周期。Java 提供 Null 指针检测、数组边界检测、异常出口、Bytecode 校验等技术，有效保证了 Java 语言的健壮性。

（6）体系结构中立：Java 解释器生成与体系结构无关的字节码指令，只要安装了 Java 运行时系统，Java 程序就可在任意的处理器上运行。这些字节码指令对应于 Java 虚拟机中的表示，Java 解释器得到字节码后，对它进行转换，使之能够在不同的平台运行。

（7）可移植性：与平台无关的特性使 Java 程序可以方便地被移植到网络上的不同机器。同时，Java 的类库中也实现了与不同平台的接口，使这些类库可以移植。另外，Java 编译器是由 Java 语言实现的，Java 运行时系统由标准 C 实现，这使得 Java 系统本身也具有可移植性。

（8）解释执行：Java 解释器直接对 Java 字节码进行解释执行。字节码本身携带了许多编译时信息，使得连接过程更加简单。

（9）多线程：多线程机制使应用程序能够并行执行，而且同步机制保证了对共享数据的正确操作。通过使用多线程，程序设计者可以分别用不同的线程完成特定的行为，而不需要采用全局的事件循环机制，这样就很容易地实现网络上的实时交互行为。

（10）动态性：Java 的设计使它适合于一个不断发展的环境。在类库中可以自由地加

入新的方法和实例变量而不会影响用户程序的执行。并且 Java 通过接口来支持多重继承，使之比严格的类继承具有更灵活的方式和扩展性。

3．丰富的类库

Java 提供了大量的类库以满足网络化、多线程、面向对象系统的需要。这些类库由 Java 开发工具包提供，当前也有很多开源的、免费的 Java 第三方类库，可以借助这些类库开发出丰富、功能强大的 Java 应用系统。

- 语言包提供的支持包括字符串处理、多线程处理、数学函数处理等，可以用它简单地实现 Java 程序的运行平台。
- 实用程序包提供的支持包括哈希表、堆栈、可变数组、时间和日期等。
- 输入输出包用统一的“流”模型来实现所有格式的 I/O，包括文件系统、网络、输入/出设备等。
- 低级网络包用于实现 Socket 编程。
- 抽象图形用户接口软件包，实现了不同平台计算机的图形用户接口部件，包括窗口、菜单、滚动条、对话框等，使得 Java 可以移植到不同平台的机器。
- 网络软件包支持 Internet 的 TCP/IP 协议，提供了与 Internet 的接口。它支持 URL 连接，WWW 的即时访问，并且简化了用户/服务器模型的程序设计。

正如 Java 的创始人之一 James Gosling 说：“Java 不仅仅只是 applets，它能做任何事情”，Dta 咨询公司的高级软件工程师 Rich Kadel 说：“Java 不仅仅是一种程序设计语言，更是现代化软件再实现的基础；Java 还是未来新型 OS 的核心；将会出现 Java 芯片；将构成各种应用软件的开发平台与实现环境，是人们必不可少的开发工具”。学好、用好 Java 语言是一个漫长的过程，需要学习大量的知识、进行大量的实践，但学好 Java 将会使你终生受益。

10.1.2　Java 语言与 P2P

在当前很多的 P2P 应用系统中，有很多都是由 Java 语言开发的，在全球最大的开源社区中，有无数的 P2P 项目也是由 Java 语言开发而成。可以说，Java 语言几乎可以实现所有的 P2P 应用。

1．P2P的Jxta解决之道

Jxta 是由 Sun 的首席科学家兼 CEO Bill Joy 提出来的；Jxta 正在被成千上万的开放源代码开发者模型化。在 P2P 领域中，Jxta 正在进行着巨大的改进。它定义了一套协议，开发者可以使用这些协议来建立几乎所有的 P2P 应用。同时，这些协议也非常灵活，可以适合不同应用的特别需要。

虽然 Jxta 也不使用特定的编程语言或者环境，不过使用 Java 无疑是一个最佳的选择，原因在于 Java 便携性、容易开发和丰富的类库，这些特性也使得 Java 借助 JXTA 平台可以开发任何 P2P 应用。

2．Java开发的功能强大的BT系统——Azureus

虽说，现在还没有准确地统计世界上使用最广泛的 BT 工具是什么，但从互联网上来自国外的一些不完全的统计数据上来看，Azureus 在英文用户中覆盖率还是很大的。此外，Azureus 的口碑也很不错，曾获得 Sourceforge 2006 社区选择奖，在 The Blog Joint 评选的 Top10OpenSourcePrograms 名列第 4，这是个很高的评价。Azureus 系统的启动界面如图 10.1 所示。



图 10.1　Azureus 系统的启动界面截图

Azureus 是完全由 Java 开发的一个开源的 BitTorrent 客户端工具，功能十分强大，在 http://sourceforge.net 的开源社区中可以得到 Azureus 工程的详细信息，包括文档、源代码等。

从当前开源社区的“最受欢迎”源代码工程来看，Azureus 也是名列第一，如图 10.2 所示为 sourceforge.net 上于 2009 年 9 月 1 日最受欢迎的软件排名情况。

Most Popular (overall)　　Past 7 days　All time

Rank	Project	Change
1.	Azureus - File Sharing, Internet	
2.	Smart package of Microsoft's core fonts - Desktop Environment	up 2 ↑
3.	Ares Galaxy - Chat, File Sharing	down 1 ↓
4.	eMule - File Sharing	down 1 ↓
5.	FileZilla - File Sharing, File Transfer Protocol (FTP), Networking	up 2 ↑
6.	7-Zip - Backup, Compression, Packaging	
7.	PortableApps.com: Portable Software/USB - Chat, Email, File Transfer Protocol (FTP), Browsers, Office Suites, Calendar	down 2 ↓
8.	Emule Pawcio mod	
9.	GTK+ and GIMP installers for Windows - Raster-Based	
10.	Audacity - Analysis, Capture/Recording, Editors, Mixers	up 1 ↑

More

图 10.2　Azureus 软件在开源社区的排名情况

从这个排名上足以看出 Azureus 的知名度，如果能将 Azureus 的源代码全部学懂学透，你的 Java 编程技术及对 BitTorrent 的理解将有一个质的飞跃。

3．Java开发的eMule系统——jMule

jMule 是一个基于 Java 的文件共享客户端（eDonkey 2000 网络），基于 Java 语言开发的类 eMule 客户端系统，没有 eMule 的功能强大，但也实现了 eMule 协议中的所有基本要素。使用 JMule，可以与世界上的任何人分享档案。此客户端是完全免费的，并且开放源代码。名称 JMule 中 J 指的是 Java 的意思，Mule 与 eMule 的意思一样。该项目目前还处于发展的阶段，还需要很多 Java 和 P2P 的高手参与进来。jMule 运行的界面及截图如图 10.3 所示。

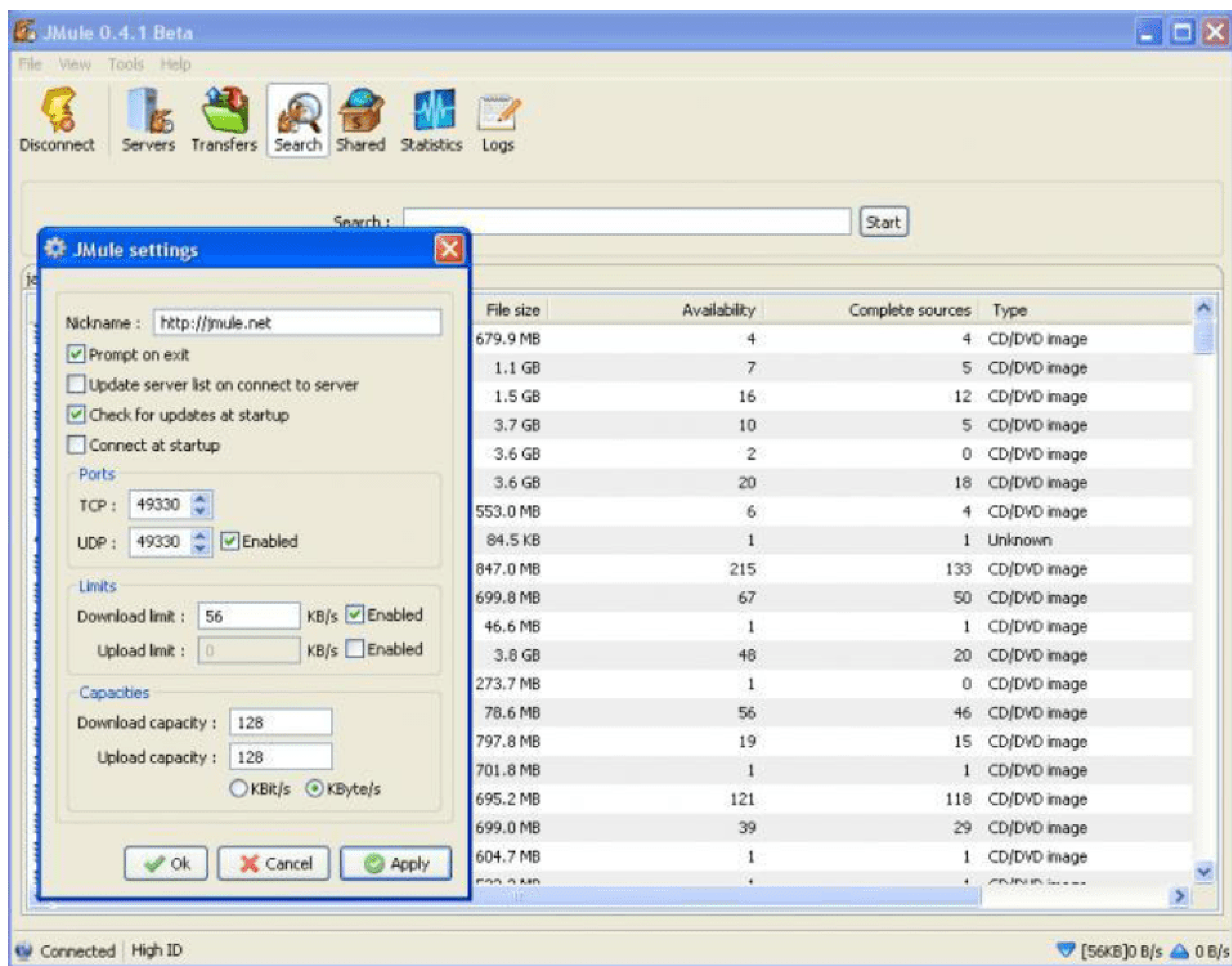


图 10.3　jMule 系统的工作界面图

从图 10.3 的界面来看，jMule 与 eMule 的整个风格、界面布局都很类似，只是功能上比 eMule 弱。JMule 是从 2008 年底开始开发的，目前版本为 0.5，能达到当前效果已经很不错了。

jMule 是在 eDonkey 2000 网路中分享文件用户的另一种选择，天然跨平台。感兴趣的读者不妨一试，它的官方网址为 http://jMule.org。

4．Java的其他P2P应用

基于 Java 的其他 P2P 应用还有很多，如 Java 开发的 P2P 即时通信系统、流媒体系统、分布式计算系统等，几乎在 P2P 应用的每个领域都活跃着 Java 和身影。正是基于此，所以本书中所有的实例和程序都是用 Java 实现的。

10.2　Windows 下 JDK 的安装与配置

本书所有的实例及代码都是基于 Java 语言实现的，要开发这些实例或是运行相关代码，需要 Java 环境的支持。本节就讲一下，如何在 Windows 系统下搭建 Java 开发平台，本章非常适合 Java 的初学者阅读。

10.2.1　关于 JDK

JDK 的全称为 Java Development Kit，Java 开发工具集的意思，进行 Java 相关开发，必需要用到这个工具集，下面要简要介绍一下 JDK 的相关知识。

1．Java的平台

Java 有 3 种开发平台分别为 J2SE、J2ME 和 J2EE。

- ❑ J2SE：是标准平台，用于桌面应用程序的开发。
- ❑ J2ME：是微型平台，用于嵌入式软件开发，如果要开发小中间件和无线应用程序，则要使用 J2ME.
- ❑ J2EE：是企业版平台，基于 B/S 结构的企业级软件开发，如果要生成服务器方程序，小服务和其他服务器方程序，则要取得 J2EE。

2．JDK的一些基本概念

关于 JDK，还有一些基本的概念需要理解。

（1）什么是 Java 2？

为什么 Java 的 3 种平台都叫 J2XX 呢？简单地说，我们现在用的 Java 是 Java 2，这是根据 JDK 的版本来划分的，1.2 以前的叫 Java 1，而此以后的就叫 Java 2。

（2）JRE 是什么？

JRE（Java Runtime Environment）是 Java 运行时环境，JDK 中包括了它，但是对于不需要开发只是运行的用户是可以只单独安装 JRE 的，所以 Sun 提供了 JRE 的下载。

3．什么是JDK

JDK，就是 Java 开发工具包，里面是 Java 类库和 Java 的语言规范，同时 Java 语言的任何改进都会加到其中作为后续版本发布。JDK 本身并不是一个像 Jbuilder 这样的开发软件，它不提供具体的开发平台，它所提供的是无论你用何种开发软件写 Java 程序都必须用到的类库和 Java 语言规范。没有 JDK，你的 Java 程序根本就不能用。

JDK 是整个 Java 的核心，包括了 Java 运行环境（JRE-Java Runtime Environment），一堆 Java 工具和 Java 基础的类库（rt.jar）。不论什么 Java 应用服务器实质都是内置了某个版本的 JDK。因此掌握 JDK 是学好 Java 的第一步。

最主流的 JDK 是 Sun 公司发布的 JDK，除了 Sun 公司之外，还有很多公司和组织都开发了自己的 JDK，例如 IBM 公司开发的 JDK，BEA 公司的 Jrocket，还有 GNU 组织开

发的 JDK 等。其中 IBM 的 JDK 包含的 JVM（Java Virtual Machine）运行效率要比 Sun JDK 包含的 JVM 高出许多。而专门运行在 x 86 平台的 Jrocket 在服务器端运行效率也要比 Sun JDK 好很多。但不管怎么说，进行基础性的 Java 开发，先把 Sun JDK 掌握好就可以了。

从初学者角度来看，采用 JDK 开发 Java 程序能够很快理解程序中各部分代码之间的关系，有利于理解 Java 面向对象的设计思想。JDK 的另一个显著特点是随着 Java（J2EE、J2SE 以及 J2ME）版本的升级而升级。但它的缺点也是非常明显的，就是从事大规模企业级 Java 应用开发非常困难，不能进行复杂的 Java 软件开发，也不利于团体协同开发。

10.2.2　JDK 的下载与安装

本节主要介绍如何从网络中下载 JDK 软件，如何在 Windows 平台下安装 JDK，这是安装 Java 开发平台的第一步。

1. 下载JDK

在 Sun 的官方网站 http://java.sun.com/javase/downloads/index.jsp 上可以免费下载 JDK 工具包、文档、源码等，也可以将已有 JDK 进行升级。打开以上网址的页面后，找到 JDK 的下载链接，如图 10.4 所示。

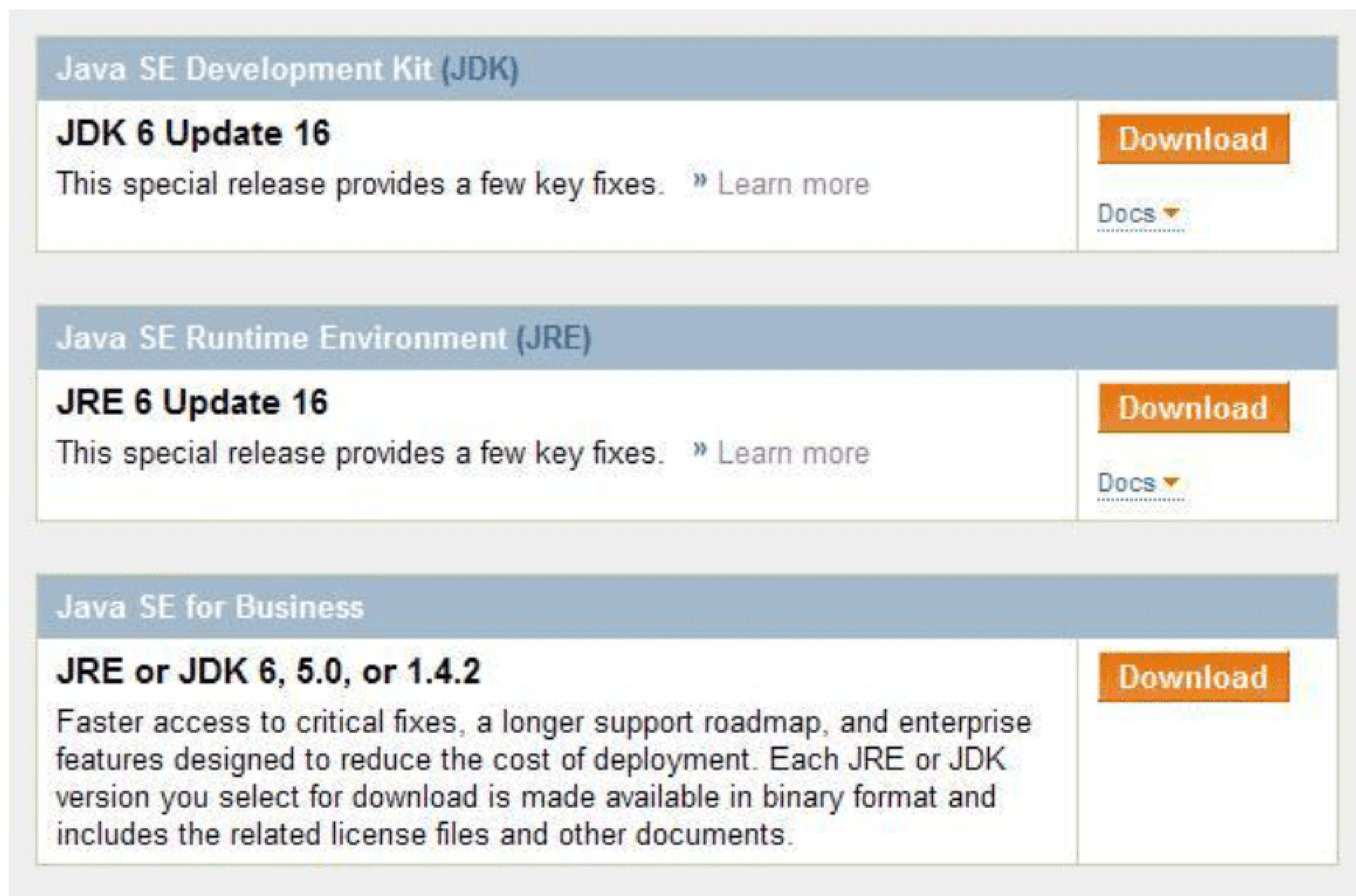


图 10.4　JDK 的下载的主页截图

图 10.4 中，单击 JDK 6 Update 16 区域中的 Download 按钮，就进入下一个页面。然后选择系统平台，这里选择 Windows，接着同意下载协议，单击 Continue 按钮后，进入 JDK 的下载页面，如图 10.5 所示。

在图 10.5 中，发现 JKD 的最新版本及全称为：“Java SE Development Kit 6u16 for Windows, Multi-language”，版本号为 JDK6u16，大小约为 73.5MB。选中相应的文件就可以进行下载了。

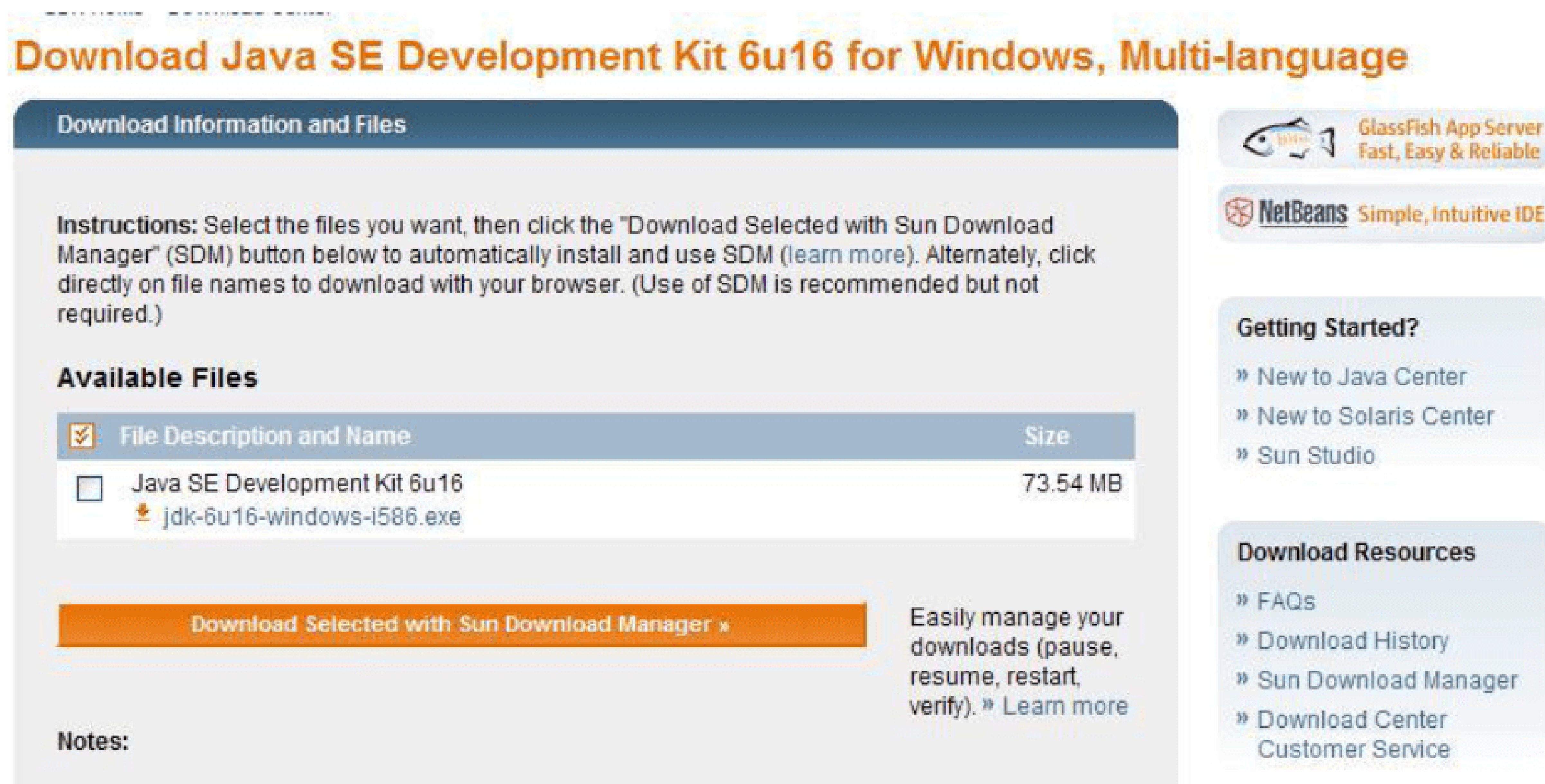


图 10.5　JDK 下载设置页面

注意： Sun 的网站上还提供了其他的有许多版本下载，本文演示用的 JDK 都是最新的版本，读者可根据需要选择适合自己的版本下载或进行升级。

2．JDK的安装

下载的 JDK 文件是个 exe 文件，直接双击就可以运行。启动 JDK 的安装程序后，首先要同意安装协议，接着会让你选择安装的软件包和安装路径，如图 10.6 所示。

在图 10.6 中，选择系统默认的安装方式，安装路径一般也不需要改动，默认的路径是 C:\Program Files\Java\JDK1.6.0_16\。直接单击“下一步”按钮就可以开始进入安装过程。在 JDK 的整个安装过程中，如果系统中没有 JRE，还会提示安装 JRE，如图 10.7 所示。

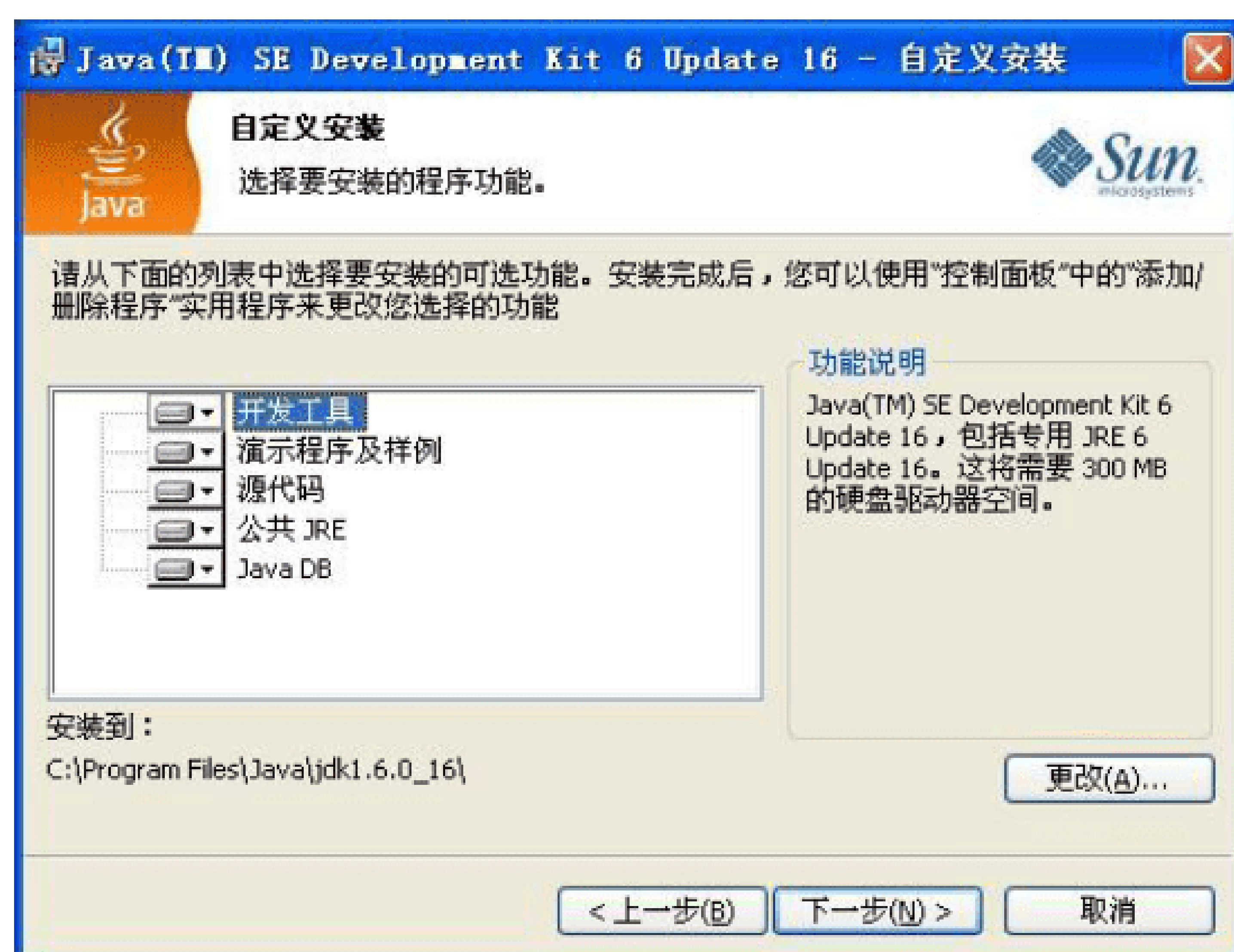


图 10.6　JDK 的安装向导界面



图 10.7　JRE 的安装提示界面

这时，仍然选择默认的设置和路径，直接单击“下一步”按钮就可以。安装过程结束后，会弹出如图 10.8 所示的提示界面，证明 JDK 的安装就完成了。单击“完成”按钮，安装过程结束。



图 10.8　JDK 安装完成界面

10.2.3　环境变量的设置

安装完 JDK 以后，还要进行环境变量的配置，这样，Java 程序就能找到已安装的 JDK 和其他配置信息了。在命令中运行 Java 程序的时候也会比较方便，如果没有其他的外部引用包时，就不需再手动载入类路径了。关于环境变量的配置方法如下。

（1）右击“我的电脑”，在弹出的快捷菜单中选择“属性”。在弹出的对话框中选择“高级”标签，然后在此界面中单击“环境变量”按钮，弹出如图 10.9 所示的环境变量设置对话框。

环境变量的设置过程中，有 3 个参数需要配置，分别为 CLASSPATH、JAVA_HOME 和 Path。Path 变量是 Windows 系统本来就有的无须新建，只需添加相应的值即可，而 CLASSPATH 和 JAVA_HOME 变量，则需新建变量名后再设置相应的值。

（2）在图 10.9 所示的环境变量设置界面中，上半部分显示是用户变量，以当前登录主机的用户名来标识，只对当前用户有效。下半部分是系统变量，所设置的环境变量值对所有用户都有效。如果希望所有用户都能使用，就在系统变量下单击“新建”按钮，在“变量名”文本框中输入 JAVA_HOME，“变量值”文本框中输入 JDK 的安装目录，如图 10.10 所示。

注意： JAVA_HOME 是 JDK 的安装目录，许多依赖 JDK 的开发环境都靠它来定位 JDK，所以必须保证正确无误。本例中 JDK 的安装目录系统默认的为 C:\Program Files\Java\JDK1.6.0_16。

（3）设置完 JAVA_HOME 变量的值以后，接着设置 Path 变量的值。找到系统变量 Path，单击“编辑”按钮，在显示的“变量值”文本框中，不要改动原有的设置值，只是在值的最后附加上 JDK 和 JRE 可执行文件的所在目录即可，附加的变量值为：

```
%JAVA_HOME%\bin;%JAVA_HOME%\JRE\bin;
```



图 10.9　Windows 下环境变量设置界面



图 10.10　JAVA_HOME 的变量值设置界面

将此值附加到 Path 中的时候，多个值之间要以分号“；”隔开，而且分号要在英文状态下输入，如图 10.11 所示。

注意：如果系统安装了多个 Java 虚拟机，比如安装了 Oracle 数据库就有自带的 JDK1.3，此时就必须把当前的 JDK 1.6 的路径放在其他 JVM 的前面，否则 Eclipse 启动时将会报错。

（4）最后一个需要设置的环境变量是 CLASSPATH，Java 虚拟机在运行的时候，会根据 CLASSPATH 设定值来搜索 class 字节码文件所在目录，但这不是必须的。可以在运行 Java 程序时显式的指定 CLASSPATH，比如在 Eclipse 中运行写好的 Java 程序时，它会自动设定 CLASSPATH，但是为了在控制台能方便地运行 Java 程序，建议最好还是设置一个 CLASSPATH。

设置 CLASSPATH，需要在系统变量里新建一个名为 CLASSPATH 的变量名，再将 JDK 的一些常用包、类库所在的路径设置为它的值即可。此处 JDK 的安装路径 CLASSPATH 的值为：

```
.;%Java_HOME%\lib;%Java_HOME%\lib\tools.jar; %Java_HOME%\lib\dt.jar;
```

设置方法如图 10.12 所示。



图 10.11　PATH 变量的设置界面



图 10.12　CLASSPATH 变量的设置界面

注意：CLASSPATH 值中，有多个值要以分号“;”隔开，其中有一个值为“.”，它是一个点“.”代表当前目录的意思。用惯了 Windows 的用户可能会以为 Java 虚拟机在搜索时会搜索当前目录，其实不会，这是 UNIX 中的习惯，出于安全考虑。许多初学 Java 的朋友兴冲冲地照着书上写好了 Hello World 程序，运行时却弹出 java.lang.NoClassDefFoundError，其实就是没有设置好 CLASSPATH，只要添加一个当前目录“.”就可以了。

按以上的步骤操作完毕后，整个 Java 的环境变量也就设置完成了。

10.2.4　JDK 的测试

安装完 JDK 和设置完成环境变量后，对 JDK 可以进行简单的测试，以验证 JDK 安装的正确与否。

（1）在主机窗口中，选择“开始”|“运行”命令，在弹出的“运行”对话框中输入 cmd，打开命令行界面。输入 java-version，如能出现 java 版本的提示信息（如图 10.13 所示的界面），说明安装配置正确。

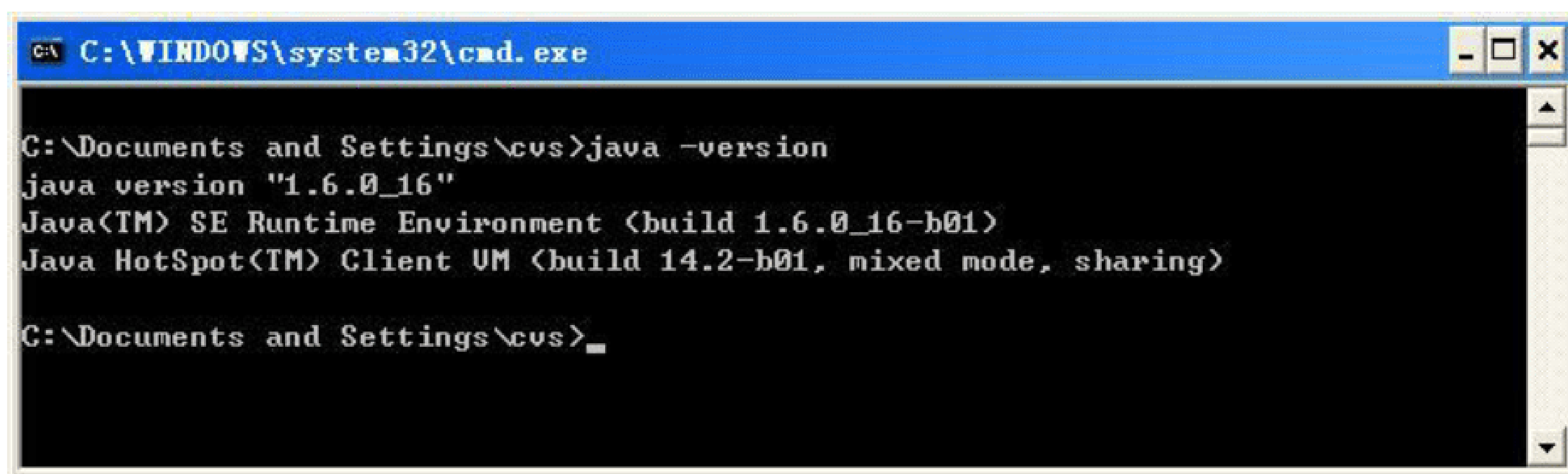


图 10.13　JDK 的测试界面

（2）JDK 安装后，下面编写一个简单的程序测试一下，检验 JDK 能否正常地编译和运行。用文本编辑器编写一个简单 Hello World 的程序。代码如下：

```
public class Hello World{
    public static void main(String  args[]){
    System.out.println("Hello World!");
    }
}
```

这就是著名的“Hello World”应用程序，保存为 Hello World.java，放在一个指定的路径下，如 D：\Hello World.java。注意 Java 是大小写相关的，不要拼写错误，否则会编译报错。

（3）接着执行编译命令。打开 cmd 命令行界面，定位到 Hello World.java 程序文件所在的目录下（本例中是 D 盘的根目录）。使用命令 javac Hello World.java，对 Java 源程序文件进行编译。

如果不报任何错误，证明程序正确，编译完成后会在源文件目录下（HelloWorld.java 程序文件所在的目录）生成一个.class 文件，也就是类文件，证明编译成功。

（4）接着执行运行命令，在 cmd 命令行界面下，用 java Hello World 命令运行编译的.class

文件，运行 Java 源程序，显示执行结果，如图 10.14 所示。

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [版本 5.1.2600]
(C) 版权所有 1985-2001 Microsoft Corp.

C:\Documents and Settings\cvs>java -version
java version "1.6.0_16"
Java(TM) SE Runtime Environment (build 1.6.0_16-b01)
Java HotSpot(TM) Client VM (build 14.2-b01, mixed mode, sharing)

C:\Documents and Settings\cvs>d:

D:\>javac HelloWorld.java

D:\>java HelloWorld
Hello World!

D:\>
```

图 10.14　Java 源程序运行界面图

这样，通过测试证明 JDK 安装成功，可以进行 Java 的相关开发了。

10.3　Eclipse 的安装与使用

JDK 只是一个 Java 开发环境，虽然普通的文本编辑工具就可以进行 Java 的开发，但一些大型的 Java 项目则需借助一些集成的 Java 工具。当前 Java 的开发工具很多，各有特长，在不同的场合有不同的应用，最常用的 Java 集成开发工具有 Eclipse、NetBeans、JBuilder、IntelliJ 等，本文中的案例都是在 Eclipse 下开发的，所以重点讲一下 Eclipse 的安装和使用方法。

注意： Java 有很多集成开发平台，根据应用和侧重点的不同，它们都有不同的特点。任何一种集成开发工具都可以开发、运行本书中的案例，所以，读者可根据自己的开发习惯选择自己喜爱的开发工具。

10.3.1　Eclipse 的下载与安装

Eclipse 是一个开源、免费的集成开发工具，是 Java 开发中的主流开发工具之一，熟练使用该工具将在学习，以及以后的实际开发中让你如虎添翼。如果把程序员类比成军队中的士兵，那么集成开发工具就是你手中的枪，你要对它足够地熟悉，并且足够熟练地使用它。

注意： Eclipse 功能强大的地方，在于它有丰富的插件，配合插件可以作为 j2ee、c、c++、.net 等多种语言的开发工具。对于开发工具的学习，需要在学习中使用，在使用中学习。

1．Eclipse的下载

Eclipse 的安装程序可以从其官方网站上免费下载，地址为 http://www.eclipse.org，进行 Eclipse 的官方网站主页后，找到其 Download 页面，如图 10.15 所示。

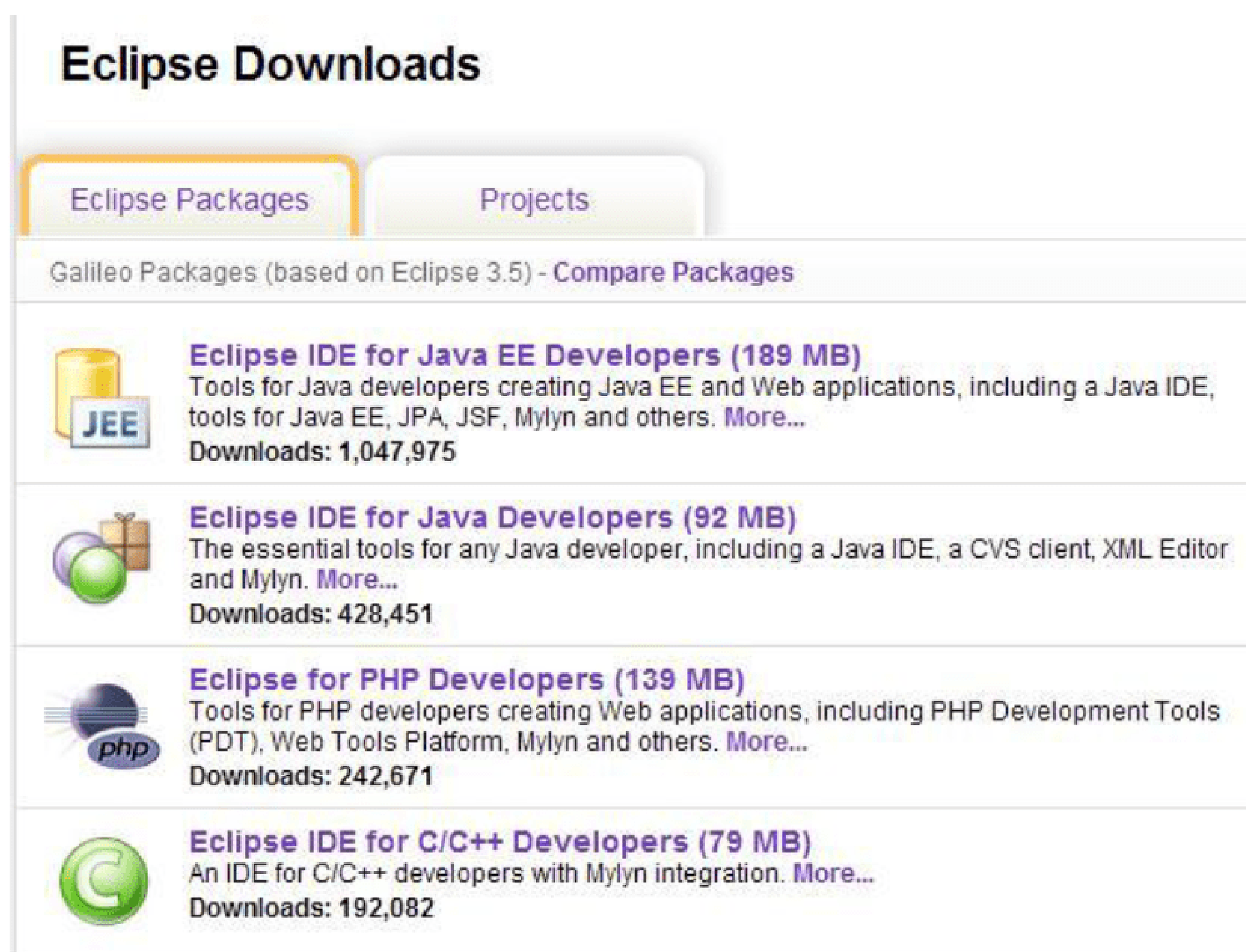


图 10.15　Eclipse 的下载页面截图

在图 10.15 所示的页面中，选择 Eclipse Classic 下载（在此网页的最下方，图 10.15 中没有显示），其最新版本为 3.5.0，如图 10.16 所示。

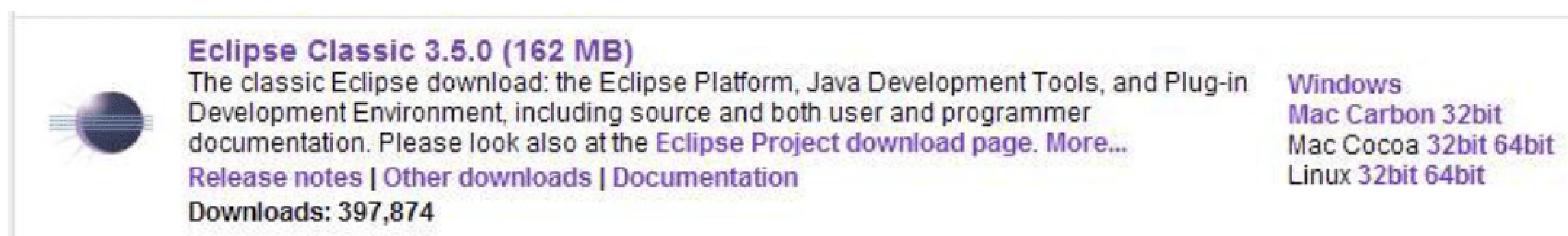


图 10.16　Eclipse Classic 3.5.0 的下载界面

根据自己的操作系统平台，直接选中下载即可。

2．Eclipse的安装

Eclipse 是一个使用 Java 语言开发的工具软件，所以在安装 Eclipse 以前，一定要安装 JDK（上文已经介绍过 JDK 的安装方法），其中 Eclipse 3.5 要求安装的 JDK 版本在 1.5 及以上。以下为 Windows 操作系统为例子来介绍 Eclipse 的安装。

Eclipse 的安装很简单，只需要解压缩安装文件即可。解压缩的文件没有限制，可以根

据实际使用的需要解压缩到任意路径下。

下载完 Eclipse 后，其完整的文件名为 eclipse-SDK-3.5-win32.zip。解压缩此压缩包到一个目录，假如解压缩到 D 盘根目录下，则会生成一个 F:\eclipse 文件夹，这个 eclipse 的文件夹就是其主程序所在的文件目录。如图 10.17 所示，是 eclipse 压缩包解压后文件结构。

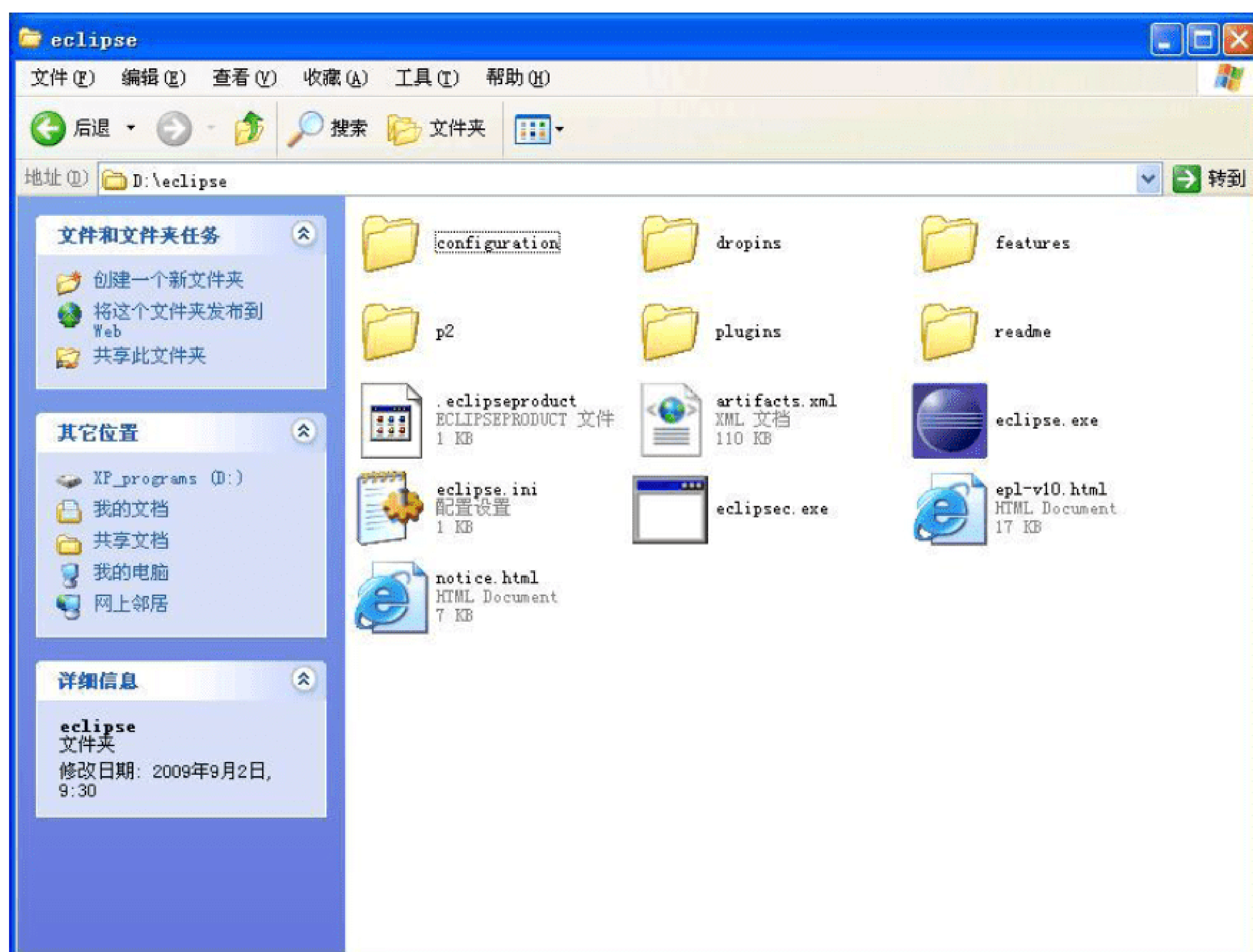


图 10.17　Eclipse 软件包的文件结构

解压后的 Eclipse 软件包，无须进行其他的配置和安装，运行 D:\eclipse\eclipse.exe 即可启动 Eclipse。

10.3.2　Eclipse 的插件安装方法

Eclipse 功能十分强大的地方，就在于它可以支持大量的插件，插件的使用及安装方法都十分灵活，本节就简要地介绍一下 Eclipse 中的插件安装方法。

Eclipse 本身有一个目录规范，从 Eclipse 的目录结构就可以看出，它主要有这样几个文件夹 eclipse、eclipse\features、eclipse\plugins，Eclipse 的插件安装一般有两种方式，一种是移植性安装、另一种是链接式（Links）的安装。

注意：Eclipse 的插件安装方法并没有严格的区分，这里是为了让读者理解方便，所以笔者在此定义了两个名字，读者只需理解这一安装过程就可以了。

1．移植性安装

移植性安装，顾名思义，就是将一个地方的插件移到另一个地方变成 Eclipse 的插件。它的安装过程就是简单地将插件中 eclipse\features、eclipse\plugins 复制到 eclipse 安装目录中的 eclipse\features、eclipse\plugins 下即可。

很多读者可能刚开始的时候不习惯 Eclipse 的英文环境，可以下一个中文语言包，将这个包以插件的方式安装到 Eclipse 中，就可以获得中文语言支持了。下面以 Eclipse 的中文语言包为例，说明一下移植性插件安装的方法。

Eclipse 的中文语言包的名称为 NLpack1-eclipse-SDK，通过普通的搜索引擎都可以搜索到相关的下载链接，也是免费软件，可以直接将其下载到本地。NLpack1-eclipse-SDK 是一个压缩包文件，下载完成后直接解压缩到一个目录，然后复制其中 plugins 目录下的所有文件和文件夹到 Eclipse 的目录 D:\eclipse\plugins 下，复制其中 features 目录下的所有文件和文件夹到 D:\eclipse\features。安装完成后，重启 Eclipse 即可。

这种安装方式有个严重缺陷，就是安装后，实际上是不可以卸载，安装过程不可逆转，无法灵活配置管理所安装的插件。

2. 链接式安装（links安装方法）

链接式安装，也有人称为 links 安装方法，它也分两种形式，一种是绝对路径安装方法，一种是相对路径的安装方法。首先看绝对路径的安装方法。

首先将中文语言包 NLpack1-eclipse-SDK 插件解压缩到一个地方，假设为 D:\myplugins 目录，解压后文件夹的目录结构如图 10.18 所示。

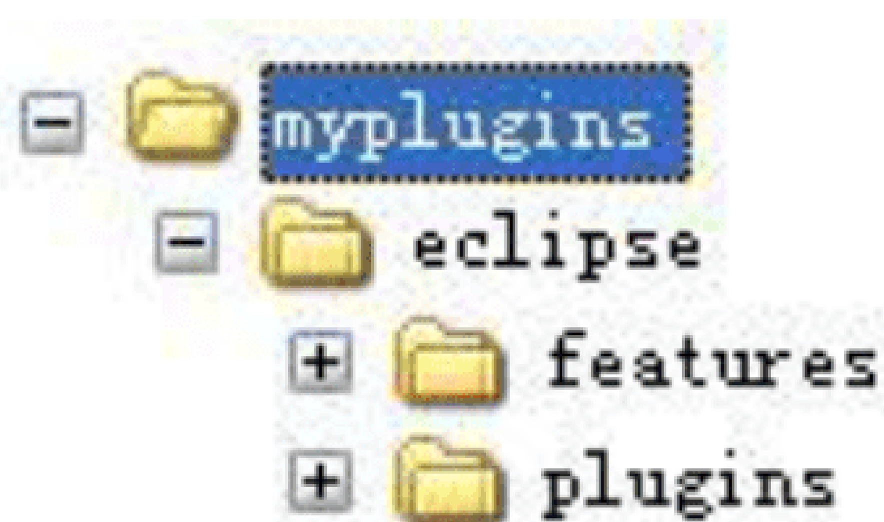


图 10.18　NLpack1-eclipse-SDK 软件包的文件结构

然后在 eclipse 的安装目录下新建一个文件夹 links，再新建一个文本文档 NLpack1-eclipse-SDK-3.2.1-win32.txt，内容如下：

```
path=D: /myplugins/NLpack1-eclipse-SDK-3.2.1-win32。
```

注意：路径的间隔符为“/”或“\\”。

根据以上操作过程，这个中文语言包的插件就安装好了。以上说的是绝对路径的 Links 插件安装方法，下面看如何变绝对为相对，使 eclipse 的运行不再依赖绝对路径，这正是相对路径安装的优点。

在上绝对路径安装过程中，我们在 Eclipse 安装目录 D:\eclipse 下面新建一个 links 和 myplugins 文件夹，将中文语言包插件 NLpack1-eclipse-SDK-3.2.1-win32.zip 解压缩到 D:\eclipse\myplugins 下，目录结构不变。

然后在 links 目录中修改新建一个文件 NLpack1-eclipse-SDK-3.2.1-win32.txt，内容如下：

```
path=myplugins/NLpack1-eclipse-SDK-3.2.1-win32
```

这样，相对路径的 Links 方法的插件安装也就完成了。

注意： 相对路径安装需要注意一点就是插件的目录 myplugins 一定要建在 eclipse 的安装目录下面，可任意命名，所有的插件最好都放到这个目录（方便管理），每个插件对应一个 links 目录里的一个文本配置文件。links 目录的名字只能是命名为 links。

不管哪种方式安装完插件以后，都需要重新启动 Eclipse，插件的功能或者界面才能显示出来。有的时候，重新启动 Eclipse 后，插件仍不能正常显示，这时候的一种解决方法是，在命令行中进入到 Eclipse 的可执行的.exe 文件目录下，在命令行中启动 Eclipse，在启动 Eclipse 的时候加上参数，-clean，例如 D:\eclipse\eclipse.exe –clean，启发后就能看见新安装的插件了。

10.3.3　Eclipse 的初始配置

Eclipse 安装完成以后，选择 Eclipse 安装目录下的 eclipse.exe 即可启动该软件。下面从 Eclipse 的启动开始，介绍一下如何使用 Eclipse。

1．工作空间设置

第一次启动 Eclipse 时，会弹出一个标题为 Workspace Launcher 的窗口，该窗口的功能是设置 Eclipse 的 workspace（工作空间），如图 10.19 所示。

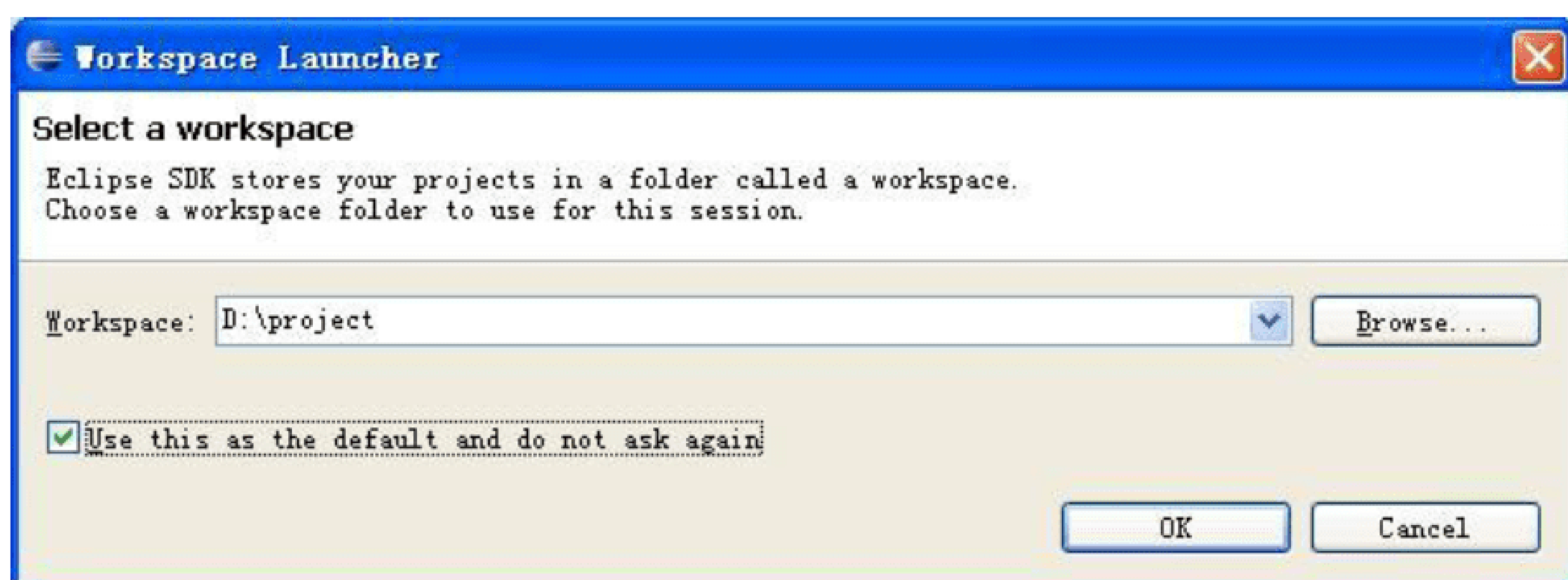


图 10.19　Eclipse 中设置工作空间的界面

workspace 是指 Eclipse 新建的内容默认的保存路径，以及 Eclipse 相关的个性设置信息。该窗口中 Workspace 输入框中是需要设置的路径，可以根据个人的需要进行设置。

注意： Workspace 是进行项目开发时源文件存放的地方，建议读者选择一个稳定、安全的位置保存此目录。

下面的 Use this as default and do not ask again 选择项的意思是，使用这个作为默认设置，以后不要再询问，选中以后的效果如下：

- 下次启动时不再弹出该窗口。
- 把这个设置作为默认设置，选中该选择项则每次启动时就不会弹出该窗口。
- 设置完成以后，单击 OK 按钮，就可以启动 Eclipse 了。

2．显示主界面

Eclipse 第一次启动后，会显示一个欢迎界面，如图 10.20 所示。

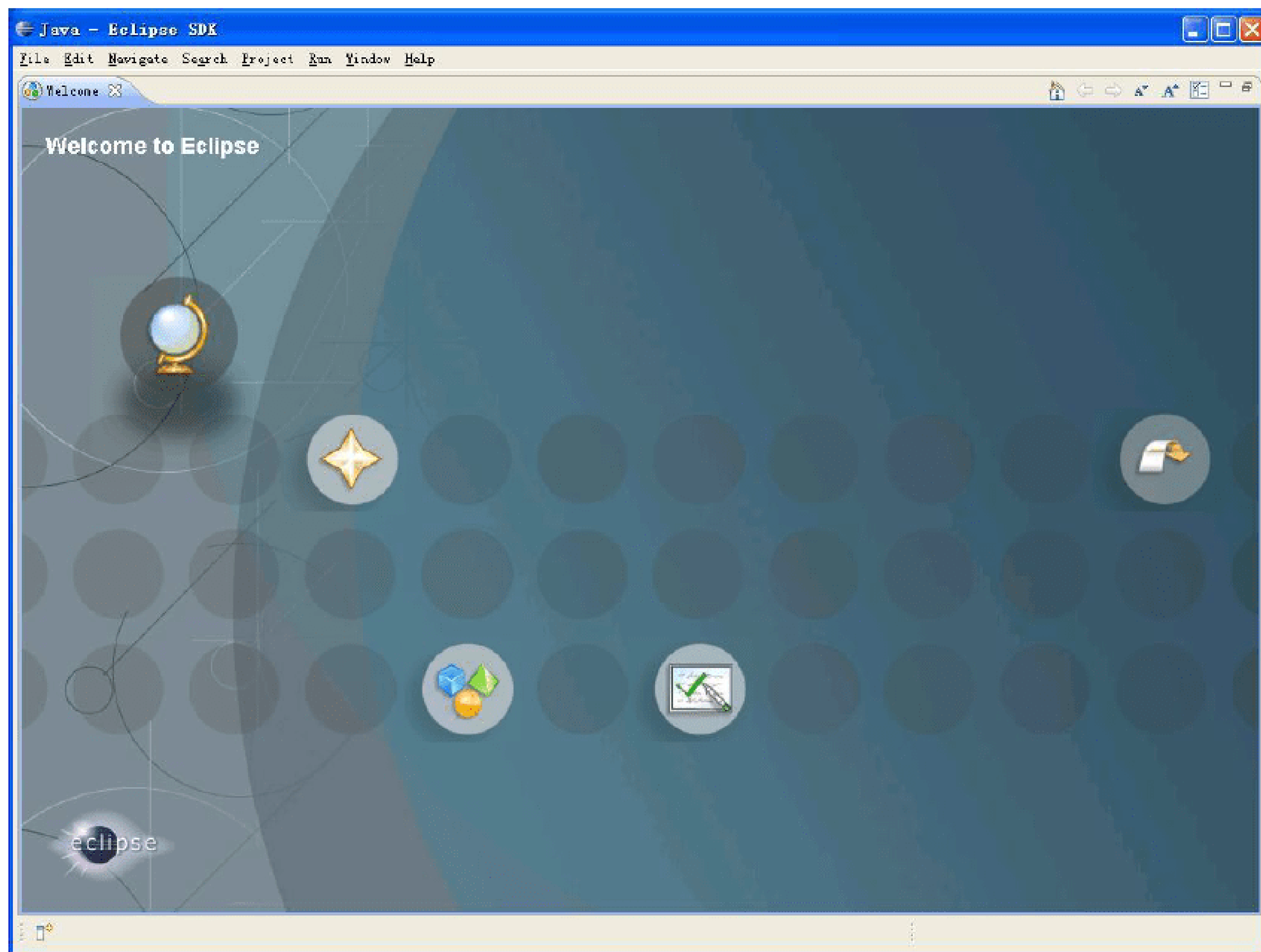


图 10.20　Eclipse 启动后的欢迎界面

在图 10.20 中，选择 Welcome 左上角的 X 符号，就可以关闭欢迎界面，接着就可以进入 Eclipse 的主界面了。Eclipse 的工作台界面如图 10.21 所示。

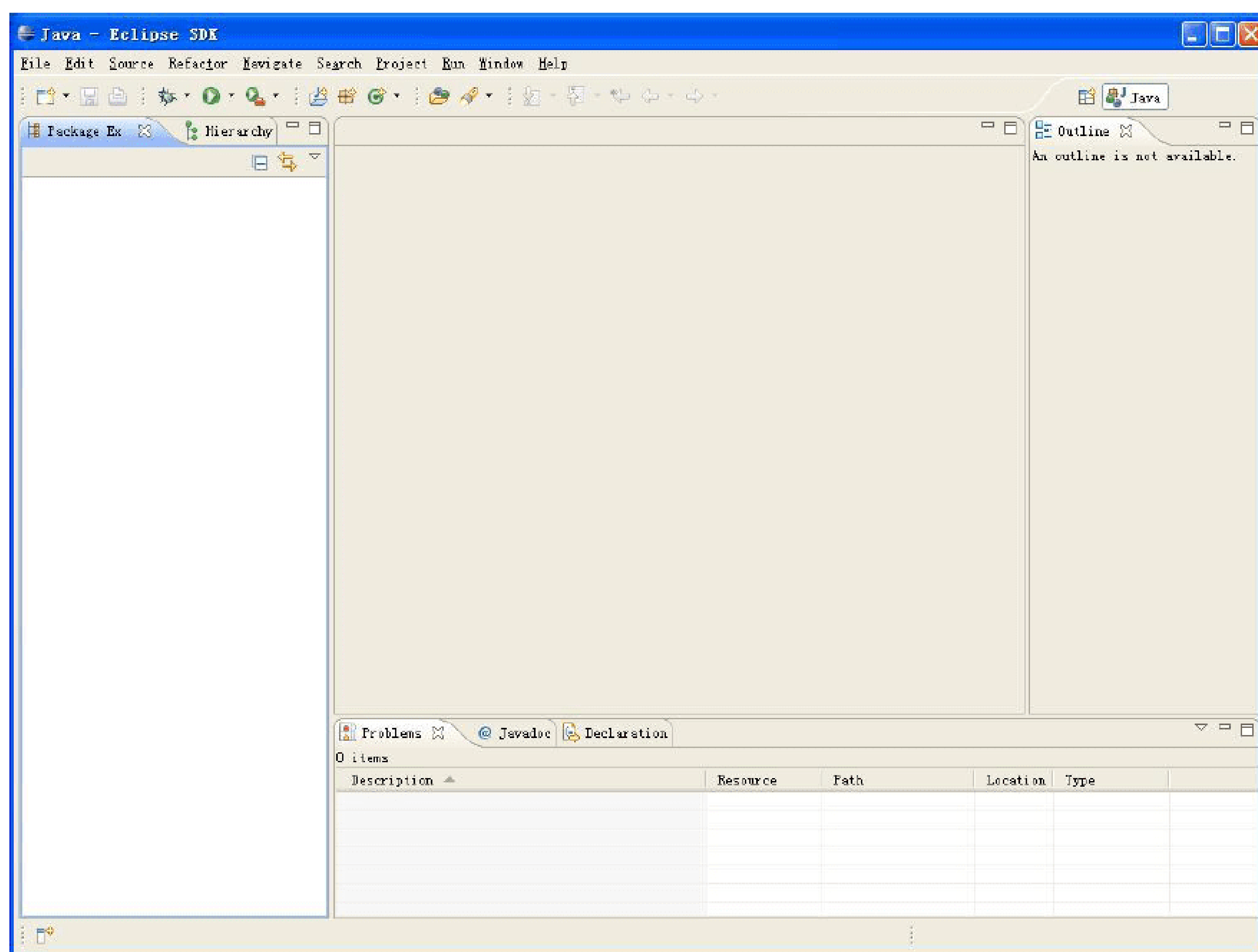


图 10.21　Eclipse 的工作主界面

欢迎界面只显示一次，以后只有在变更了工作空间以后才可能会再次显示。进入 Eclipse 主界面以后，就可以进行各种不同的 Java 项目开发了。

注意： 关于 Eclipse 界面的布局方式，会用到一些专业术语，这里就不再说明，而且 Eclipse 的布局方式很灵活、简单，多试几次、慢慢使用就会逐渐熟悉了。

10.3.4　使用 Eclipse 进行 Java 开发

集成开发环境（IDE）的使用相对来说稍显繁琐，但是对于实际的项目开发来说却是非常实用的。在初次使用时，需要习惯和适应这种使用方式。

集成开发环境在使用前，需要首先建立 Project（项目）。Project 是一个管理结构，管理一个项目内部的所有源代码和资源文件，并保存和项目相关的设置信息。

一个项目内部可以有任意多个源文件，以及任意多的资源。

使用 Eclipse 的开发 Java 工程，基本步骤如下：

（1）新建项目。

（2）新建源文件。

（3）编辑和保存源文件。

（4）运行程序。

下面分别按这几个步骤演示一下，如何在 Eclipse 下开发 Java 程序。

1．新建项目

新建项目的步骤如下。

（1）在 Eclipse 的主界面中，在顶层菜单列表中，选择 File 选项打开下拉菜单。在其中选择 New | Java Project 命令，弹出如图 10.22 所示的新建工程界面。

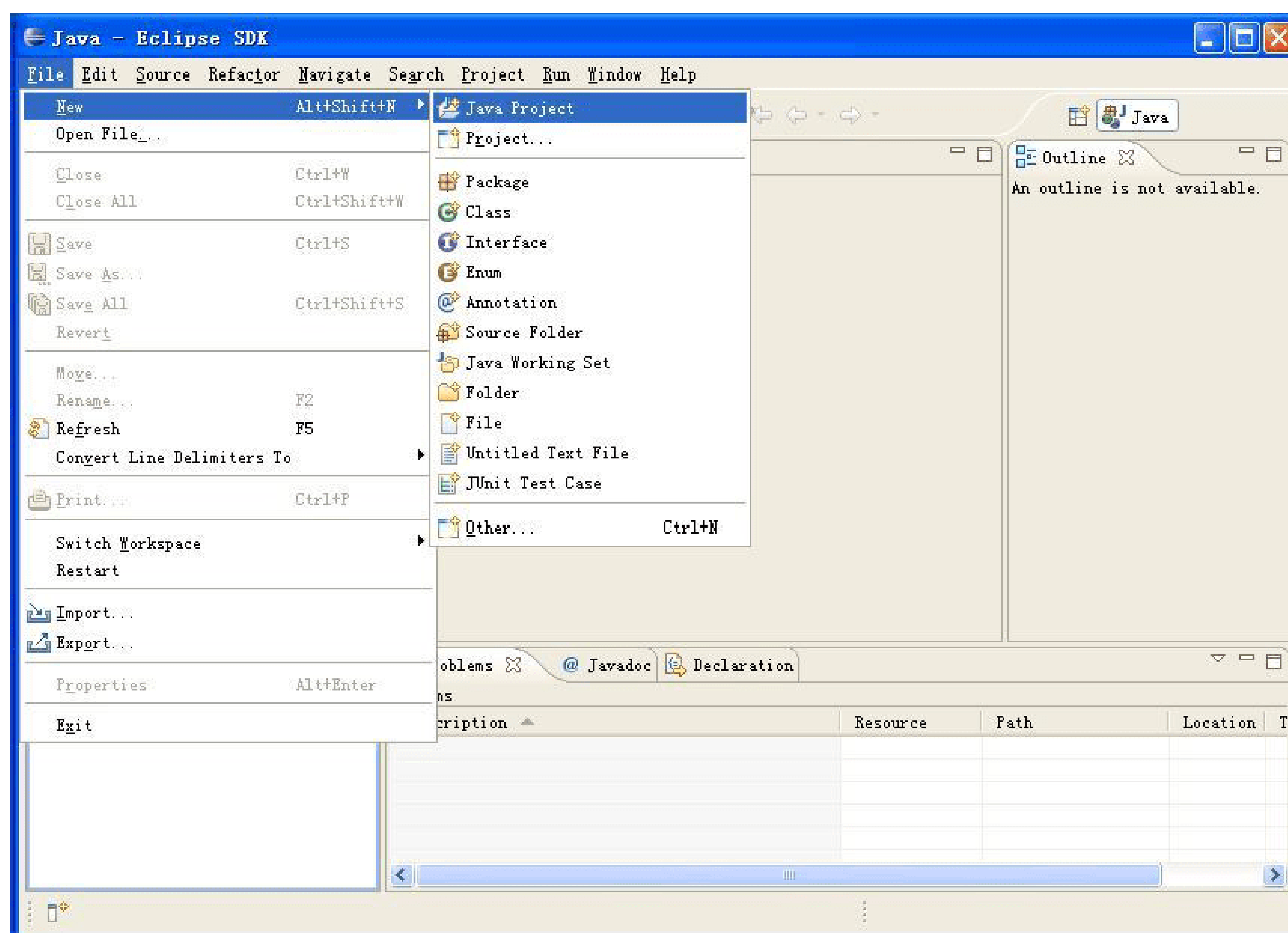


图 10.22　在 Eclipse 中新建一个项目

（2）在弹出的“New Java Project”窗口中，进行新建项目的设定，例如输入新建的工程名：myfirstproject，在这个界面的设置中，一些选项的含义如下：

- ❑ Project Name 是必须输入的内容，代表项目名称，在硬盘上会转换成一个文件夹的名称。
- ❑ Content 设置项目的内容，是在 Workspace 里新建一个新的项目，还是从已有的资源里创建，开发的时候可根据需要自行选择。
- ❑ JRE 部分设置项目使用的 JDK 版本，本例中使用的是 JavaSE-1.6 版本的，也就是 Java 6 或 JDK 1.6。
- ❑ Project layout 部分设置项目文件内部的目录结构。是分开的还是混合的，根据自己的编程习惯选择，整个设置界面如图 10.23 所示。

（3）在图 10.23 中单击 Finish 按钮完成设置，项目建立以后，可以到磁盘对应路径下观察一下项目文件夹的结构。在 Eclipse 平台下，新建完成的项目结构如图 10.24 所示。

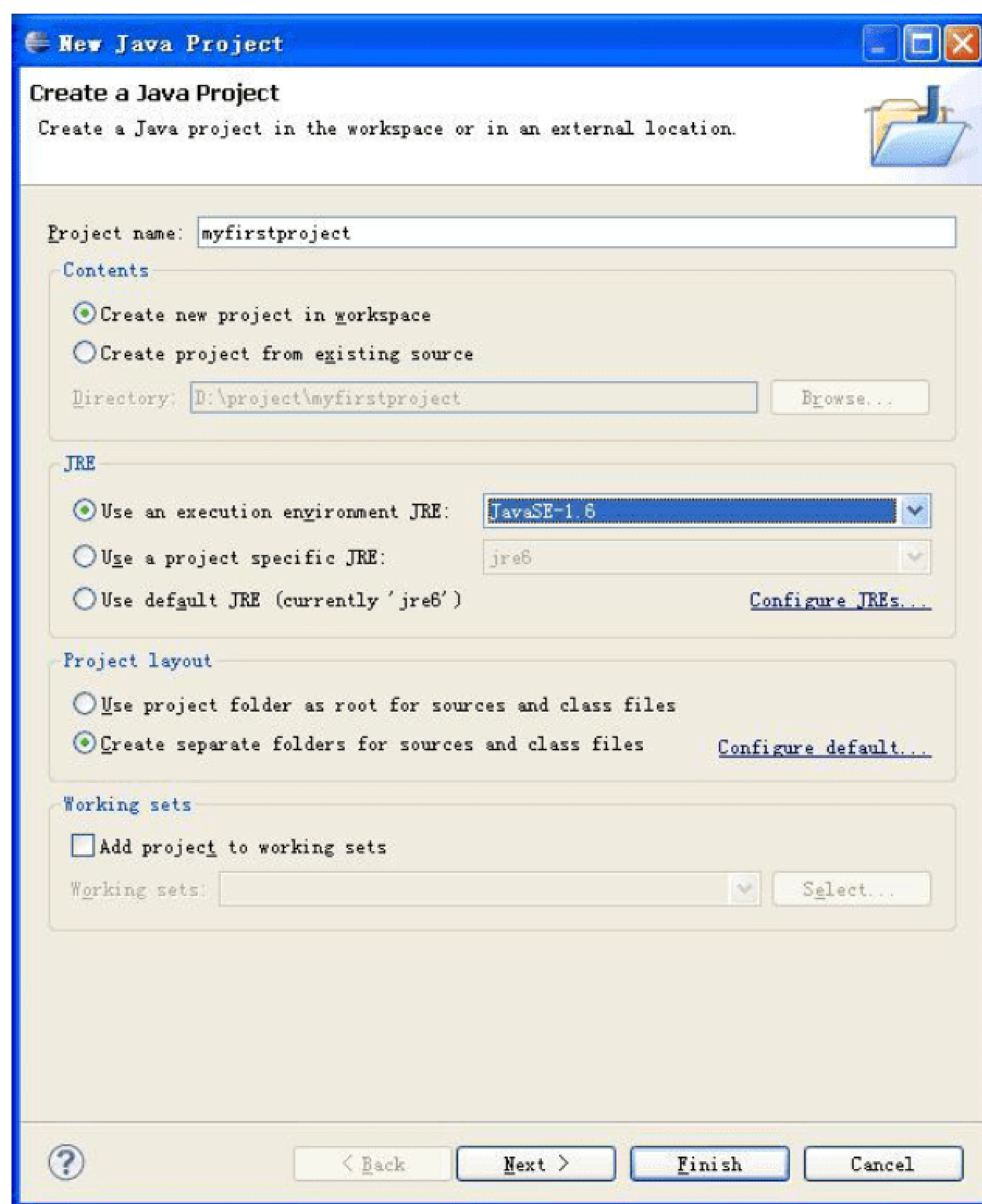


图 10.23　Eclipse 中新建工程的设置界面

2. 新建源文件

项目建立以后，或者打开项目以后，就可以新建源文件了。一个项目中可以包含多个源文件，每个源文件都可以独立执行。新建源文件的步骤如下：

在 Eclipse 的主界面窗口中，右击以上新建的工程名称，在弹出的快捷菜单中，选择 New | Class 命令，就打开新建类文件的对话框，如图 10.25 所示。

在图 10.25 所示的 New Java Class 向导界面中，可以进行新建源文件的设定，几个设定选项的含义如下：

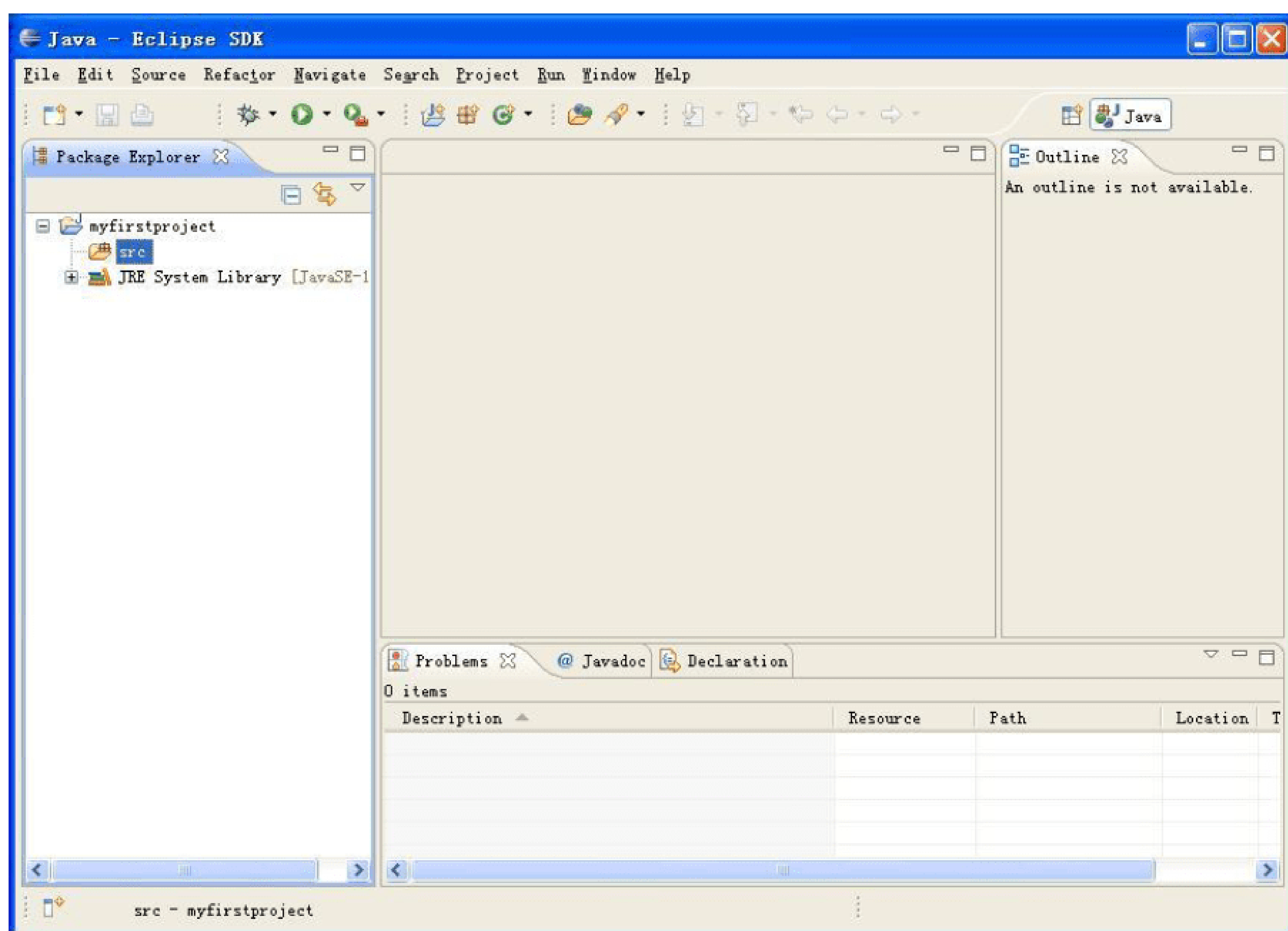


图 10.24　Eclipse 平台下新建项目后的目录结构

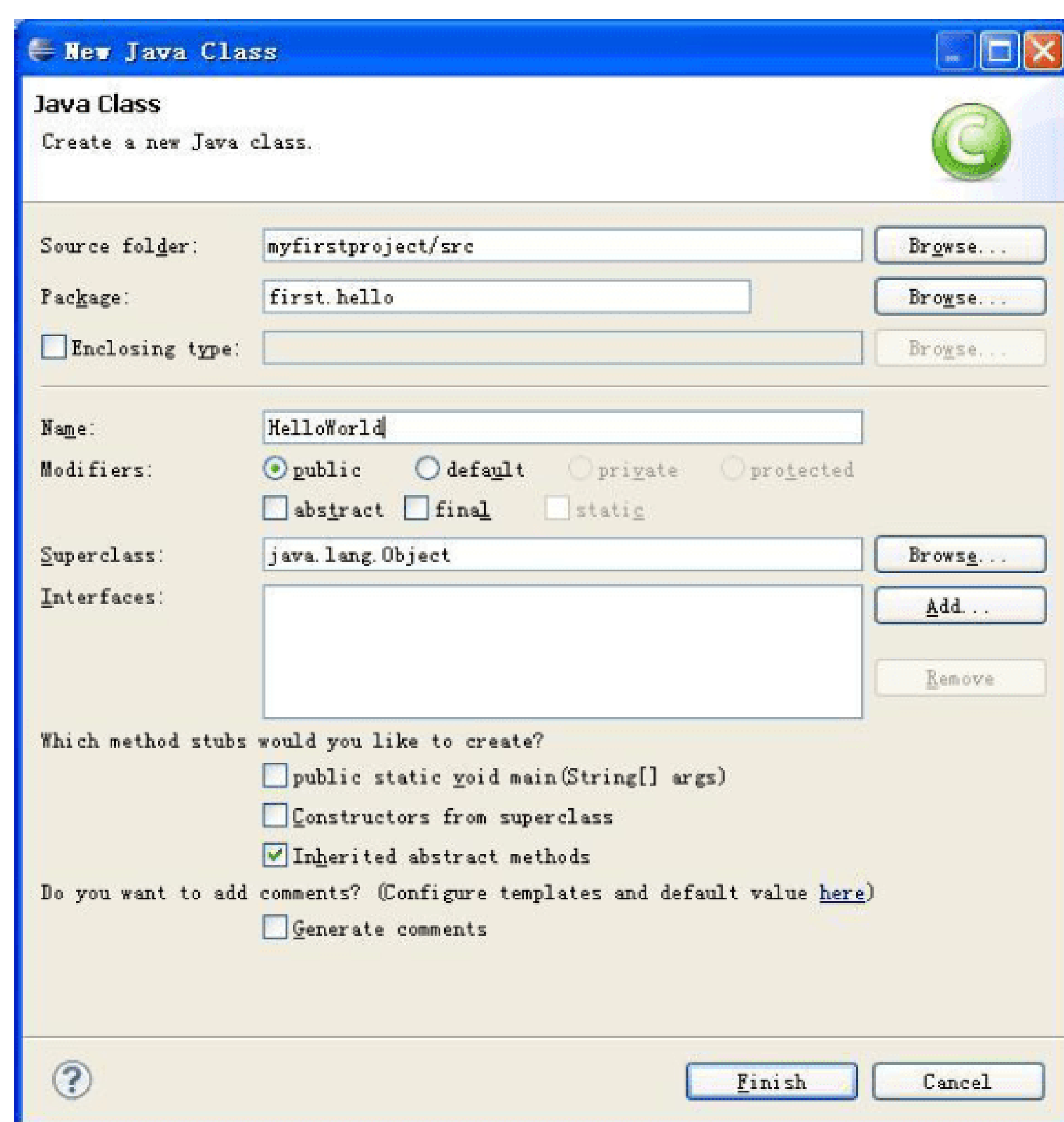


图 10.25　新建类文件向导界面

- Sourcefolder 代表源代码目录，例如 myfirstjavaproject/src，如果该内容和项目保持一致则不需要修改，否则可以选择后续的 Browse 按钮进行修改。
- Name 代表源文件的名称，例如输入 Hello World。
- public static void main（String[] args）选项代表在生成的源代码中包含该代码，可根据自己的编程习惯来选择。

设置完成以上的选项后，再单击 Finish 按钮完成设置，此时 Eclipse 将自动生成符合要求的源代码，并在 Eclipse 工作台环境中打开，如图 10.26 所示。

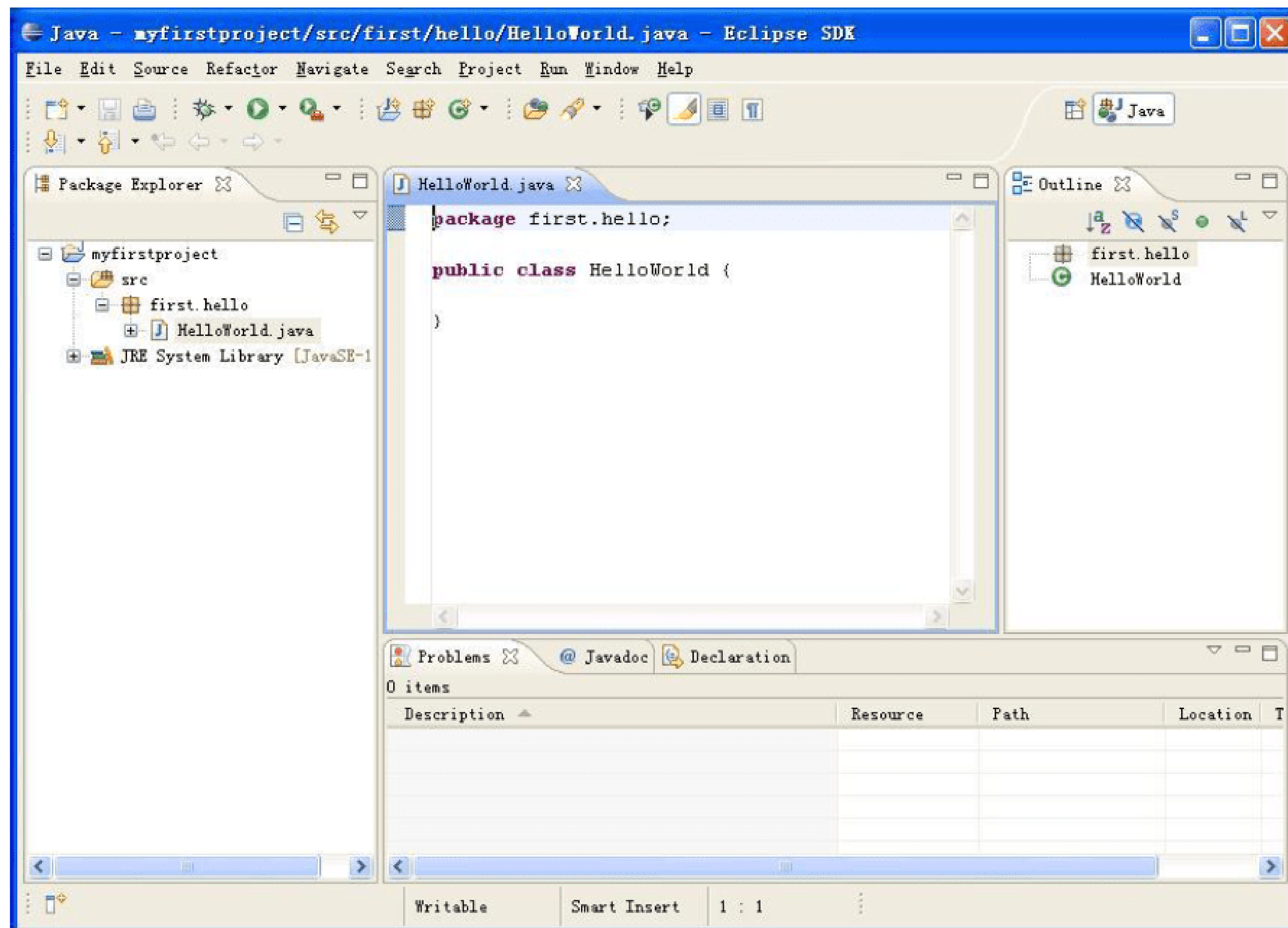


图 10.26　Eclipse 中新建 Java 源文件示意图

在图 10.26 所示的 Eclipse 工作界面中就可以编辑 Java 代码了。

3．编辑和保存源文件

在图 10.26 的工作台界面中，输入以下简单的 Hello World 测试代码，以测试 Eclipse 的运行情况，代码示例如图 10.27 所示。

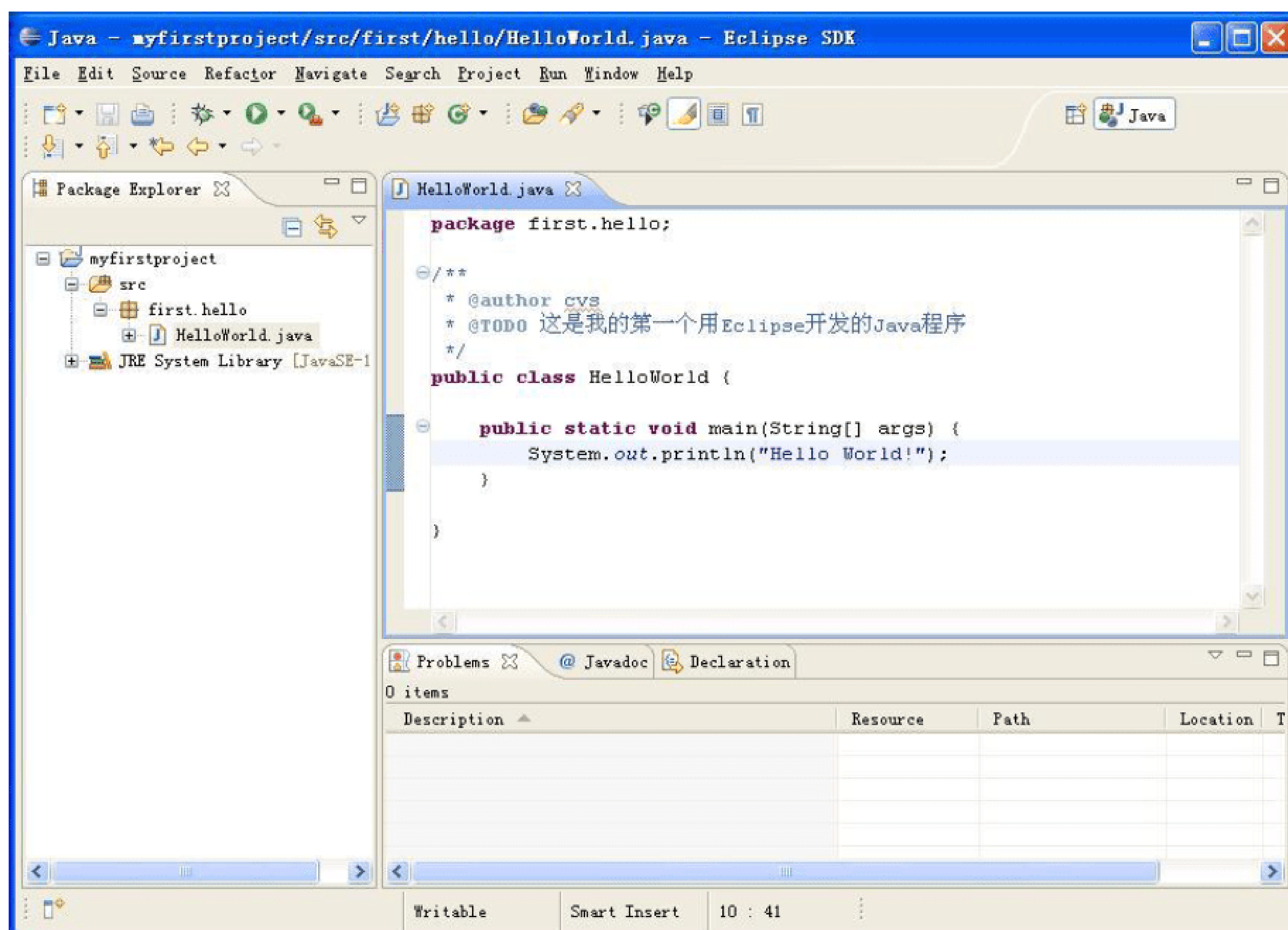


图 10.27　在 Eclipse 工作台中编辑 Java 源文件

单击工具栏的“保存”按钮，或者按下 Ctrl+S 键保存源文件。在源文件保存时，Eclipse 会自动编译该代码，如果有语法错误，则以红色波浪线进行提示。

4. 运行程序

保存完以上的源代码后，Eclipse 会自动对其进行编译，可直接运行该程序代码。运行程序的方法为：右击源代码空白处，在弹出的快捷菜单中选择 Run as | Java Application 命令即可运行。当然，也可以选择 Eclipse 左侧需要运行的文件名，右击，也可以找到一样的菜单进行运行，如图 10.28 所示。

程序的运行结果会显示在 Eclipse 的控制台界面。如果程序出现错误，要调试 Java 程序也非常简单，Run 菜单里包含了标准的调试命令，设置断点以后，和运行程序的方法一样，只是选择 Debug As | Java Application 即可，Eclipse 可以非常方便地调试 Java 应用程序。

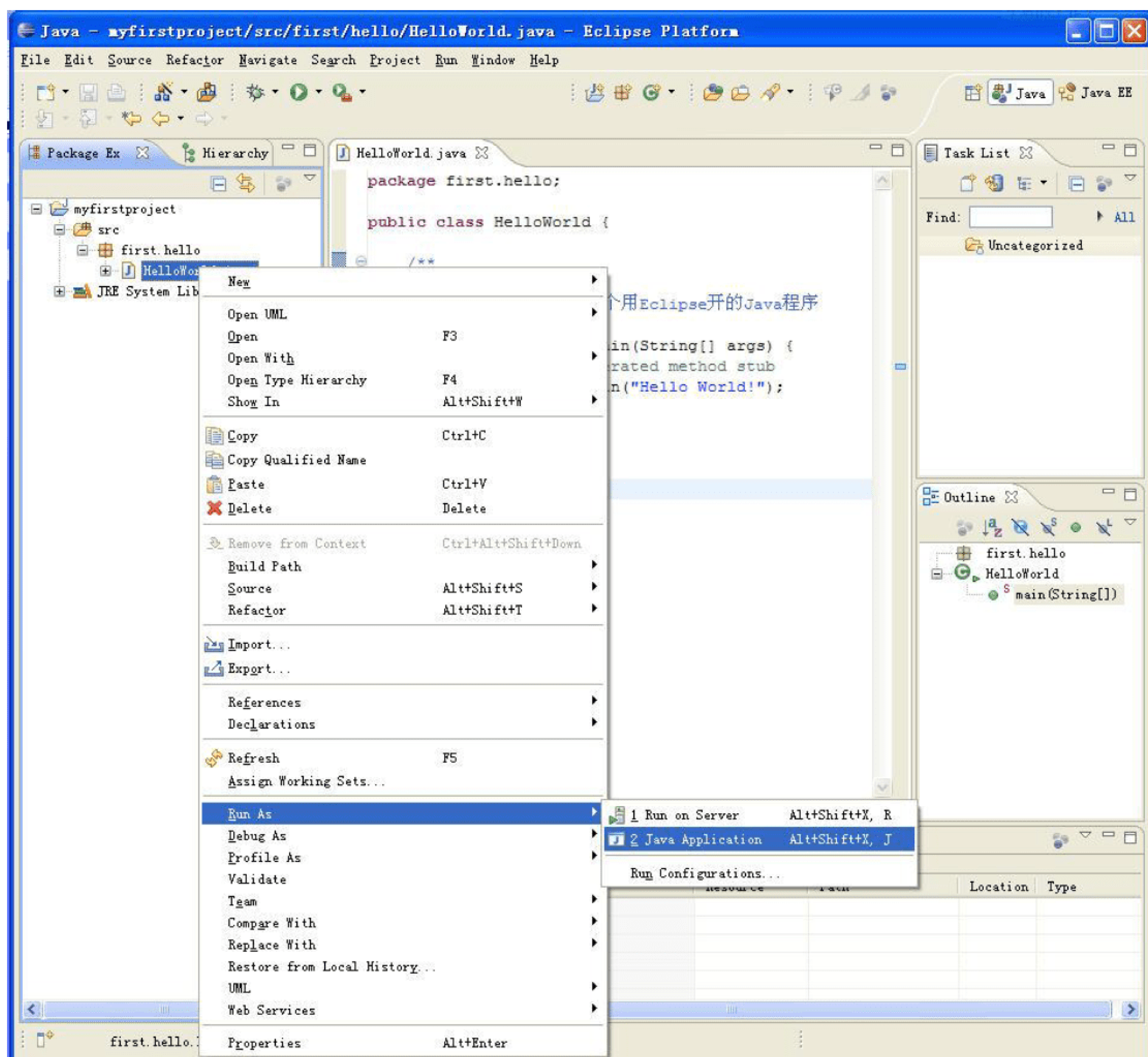


图 10.28　Eclipse 中 Java 源程序的运行方法

10.4　Linux 系统下构建 Java 开发环境

Java 是一种跨平台的语言，本书中所有的源代码都可以在 Windows 和 Linux 系统这两种平台下运行，随着使用 Linux 系统进行案例开发的读者越来越多，所以本节就讲一下如何在 Linux 下构建 Java 的开发环境。

注意：本文的所有案例都可以在 Linux 下 Java 平台中开发实现，大部分代码也都在 Linux 系统下测试通过。

10.4.1　安装 JDK 的开发环境

Linux 有很多不同的版本，本文选用的是 Linux 的 Ubuntu 发行版，内核版本为 2.6.22。

注意：关于 Linux 的 Ubuntu 发行版本及 Linux 的基本使用方法等相关知识，请读者参考其他资料，这里就不再说明。

在 Ubuntu 下的安装软件都有两种方式，一种是直接下载支持 Linux 的 JDK 软件包，进行手动安装，另一种是用命令 apt-get install 的形式安装，就是新立得安装方法（当然这种安装方法的前提是得有“源”才行）。

1．手动安装JDK 6.0

手动安装是相对于 Ubuntu 下新立得的安装方式而言的，顾名思义就是安装过程由手动操作完成，从文件的下载、复制、执行命令、信息配置等，都需要一步步的通过命令来完成，这就需要读者掌握一定的 Linux 基础和命令使用方法。

（1）从 Sun 主页下载 JDK for Linux 版本，比如当前支持 Linux 的最新版 JDK 为 JDK-6u16-linux-i586.bin，找到相应的下载链接后，直接将其下载到本地。比如下载到 /home/myfile 目录下。

（2）在 Ubuntu 下定位到下载文件的路径，用 root 用户登录 Ubuntu，或是在普通用户下用 SU 命令切换用户。切换到 JDK 将要安装的目录下。

Ubuntu 的终端里，用命令方式 cd <目录路径名>。例如，要在/usr/java/目录中安装 JDK 软件，请输入 cd/usr/java/，把刚才下载的 JDK-6u16-linux-i586.bin 文件复制到这个目录里，复制命令：

```
sudo cp /home/myfile/JDK-6u6-linux-i586.bin /usr/java
```

（3）设置权限为可执行的类型，命令如下：

```
chmod a+x JDK-6u6-linux-i586.bin;
```

（4）启动安装过程。输入./JDK-6u6-linux-i586.bin。接下来会提示二进制的许可协议，输入 yes 回车即可。安装过程如果遇到一些问题，都同样输入 yes 就可以。

（5）一路进行下去，最后看到 Done 字样，就完成了 Java 环境的安装。安装的位置就是当前目录/usr/java，当然你可以选择在别的位置。可以用 ls 命令查看一下是否正常。

2．新立得的方式安装JDK 6.0

新立得是一种 Linux 下的软件安装方式，其软件包管理器起源于 Debian。它是 dpkg 命令的图形化前端，或者说是前端软件套件管理工具。它能够在图形界面内完成 Linux 系统软件的搜寻、安装和删除，相当于终端里的 apt 命令。

在 Ubuntu 最近的长期支持版本里已经预装了新立得软件包管理器。在没有安装它的系统中，可以通过 apt-get install synaptic 进行安装。使用新立得软件包管理器的同时不能使用终端，因为它们实质上是一样的。

（1）在 Ubuntu 下用新立得的方式安装 JDK 环境，在联网的情况下在终端下输入命令：

```
$sudo apt-get install sun-java6-JRE sun-java6-sdk
```

（2）这条命令就可以帮助下载并安装 Java 6 了，顺便再给浏览器安装 Java 支持。

```
$sudo apt-get install sun-java6-plugin
```

（3）安装完这 3 个之后还需要写入系统变量：

```
$sudo gedit /etc/environment
```

（4）在文本编辑器里写入下面两行内容：

```
CLASSPATH=.:/usr/lib/jvm/java-6-sun/lib
Java_HOME=/usr/lib/jvm/java-6-sun
```

（5）还要将系统虚拟机的优先顺序也调整一下。

```
$sudo gedit /etc/jvm
```

（6）在文本编辑器里将下面一句写在最顶部。

```
/usr/lib/jvm/java-6-sun
```

这样，新立得的安装方式也完成了。

注意：在 Linux 环境下，软件的安装过程都是用命令来完成的，以上的相关命令，读者有不明白的地方，请参考 Linux 相关教程。

10.4.2　Linux 下环境变量的配置

Java 程序的运行与环境变量有紧密的关系，不管是在 Windows 下还是在 Linux 下，都要设置好环境变量，Linux 下环境变量的配置如下。

1．设置环境变量

上面安装完毕后，直接在 shell 里输入 Java 是不起作用的，需要先配置一下环境变量。设置方式是用 vi 或是 gedit 打开/etc 下的 profile 文件，命令如下：

```
sudo vi  /etc/profile
```

打开此文件后，在文件的末尾添加下面的内容。

```
#set java environment Java_HOME=/usr/java/JDK1.6.0_06;
export JRE_HOME=/usr/java/JDK1.6.0_06/JRE;
export CLASSPATH=.:$Java_HOME/lib/dt.jar:$Java_HOME/lib/tools.jar;
export PATH=$PATH:$Java_HOME/bin;
```

添加完成后，保存再退出。在 shell 终端里，可试验一下是否安装成功。echo 各个变量是否正常，如果正常，还需要进行下一步的测试。

注意：各个环境变量的值要与自己 Linux 系统下安装的路径一致。

2. 代码测试

测试 Ubuntu 中 JDK 的安装情况，只需在终端中输入命令 java –version，如果能返回如图 10.29 所示的界面提示，证明 JDK 安装成功了。



图 10.29　Linux 下 JDK 的测试

Java 环境已经建立好了以后，可以用文本编辑器编写一个 Java 的 HelloWorld 程序执行一下 javac 编译并 Java 来解释执行看看效果。如图 10.30 所示为在 Linux 下用 gedit HelloWorld.java 命令，编辑一个 Java 源文件。

保存源文件后，分别执行 Javac 编译，再执行 Java 命令运行，出现如图 10.31 所示的结果，证明 Linux 下的 JDK 安装，和环境变量的配置都成功了。

10.4.3　Linux 下安装 Eclipse

在 Windows 下 Eclipse 是直接解压就可以用的，在 Linux 下也是一样，直接下载 Eclipse 到 Linux 的本地路径下，然后解压缩，再指定快捷方式就可以了。

Eclipse 的官方下载地址为 http://www.eclipse.org/downloads/，选择 Linux 对应的最新版本即可。下载完成后，解压到指定目录，如 Linux 下的/opt 目录。

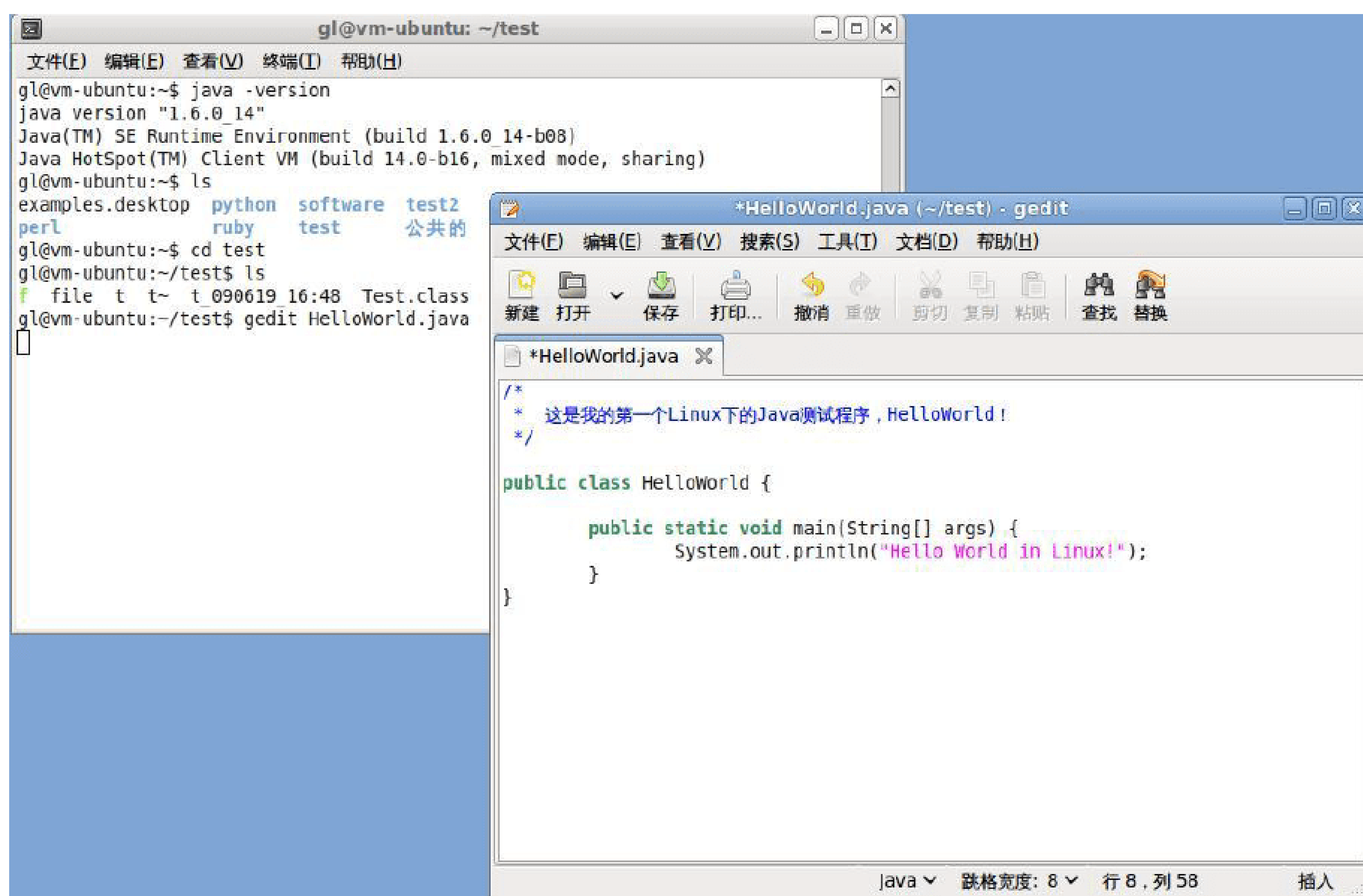


图 10.30　Linux 编辑 Java 程序



图 10.31　Linux 下 Java 源程序的运行结果界面

注意：也可先解压到当前目录，然后，再执行命令 mv eclipse /opt 移动文件到这个目录下面。

解压完成后就安装完成了，为了方便在 Linux 系统中使用 Eclipse，还需要进行进一步设置，这里只说一下启动脚本和桌面启动图标的设置方法。

1．创建启动脚本

启动脚本就是关于设置 Eclipse 启动的相关命令，在 Linux 下，大部分的操作都是靠命令来执行。有了启动脚本以后，在 Linux 的命令终端，不管是什么状态或是什么路径下，只需输入相应的启动命令，就可以很方便地启动 Eclipse。

（1）把 eclipse 目录的更改为 root 拥有，可以执行下面的命令：

```
sudo chown -R root:root /opt/eclipse
```

（2）编写启动脚本。

在/usr/bin 目录下创建一个启动脚本 eclipse，这样，在终端的任何路径下，直接输入 eclipse 命令，就可以启动 Eclipse 运行。用 gedit 编辑一个 eclipse 文件，命令如下：

```
sudo gedit /usr/bin/eclipse
```

然后在该文件中添加以下内容。

```
#!/bin/sh
export MOZILLA_FIVE_HOME="/usr/lib/mozilla/"
export ECLIPSE_HOME="/opt/eclipse"
$ECLIPSE_HOME/eclipse $*
```

（3）修改该脚本的权限，让它变成可执行，执行下面的命令：

```
sudo chmod +x /usr/bin/eclipse
```

2. 在桌面或者gnome菜单中添加eclipse启动图标

在 Linux 桌面或菜单中创建一个启动图标，这类似于 Windows 的桌面快捷方式，有了这个图标，可以在桌面或是 gnome 菜单中轻松地打开 Eclipse 应用程序，这样就非常方便使用。创建方法如下：

在桌面（右击桌面，在弹出的快捷菜单中选择创建启动器）或面板（右击面板，在弹出的快捷菜单中选择添加到面板 | 定制应用程序启动器）上创建一个新的启动器，直接右击，这时会弹出一个快捷菜单。在其中选择“创建启动器”选项，然后在弹出的对话框中，添加下列数据：

名称：Eclipse Platform；

命令：eclipse；

图标：/opt/eclipse/icon.xpm。

3. 在Applications（应用程序）菜单上添加一个图标

在应用程序菜单中添加图标，其实也是添加一个快捷方式的方法，类似于 Windows 系统中“开始“|”程序”里的应用程序菜单，有些读者并不喜欢桌面上放置太多的图标，那么应用此方法，则可以将 Eclipse 的启动图标直接指向应用程序的菜单里。具体设置方法如下。

（1）用文本编辑器在/usr/share/applications 目录里新建一个名为 eclipse. desktop 的启动器，命令如下：

```
sudo gedit /usr/share/applications/eclipse.desktop
```

（2）然后在文件中添加下列内容。

```
[Desktop Entry]
Encoding=UTF-8
Name=Eclipse Platform
Comment=Eclipse IDE
Exec=eclipse
Icon=/opt/eclipse/icon.xpm
Terminal=false
```

```
StartupNotify=true
Type=Application
Categories=Application;Development;
```

保存文件。完成整个设置过程。以上设置完成以后，就可以在 Linux 的菜单中找到 Eclipse 的链接了，如图 10.32 所示。

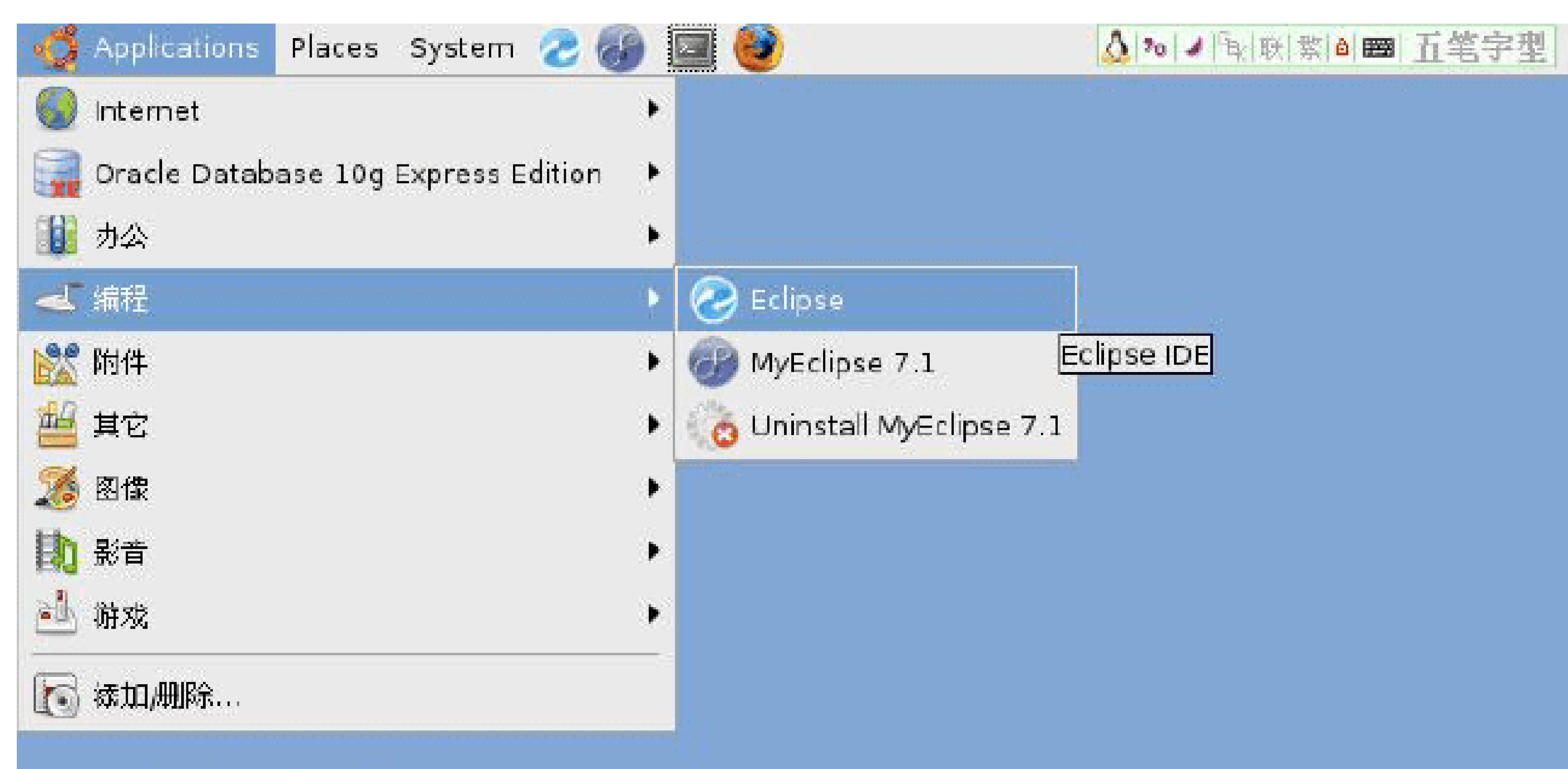


图 10.32　Linux 下 eclipse 在系统菜单下的链接

图 10.32 中，双击 eclipse 的图标就可以启动 Eclipse 运行了。如果没有报出任何错误，证明 Linux 下 Eclipse 的安装完全正确。

10.4.4　Linux 下 Eclipse＋JDK 的测试

Linux 下的 Eclipse 安装完成后，它的启动、设置、使用方法与 Windows 平台下完全一样。现在就以 Linux 下的一个 Hello Word 程序来测试 Eclipse+JDK 的安装效果。

（1）如图 10.33 所示，在 Linux 下的 Eclipse 中新建一个 Hello Word.java 文件。

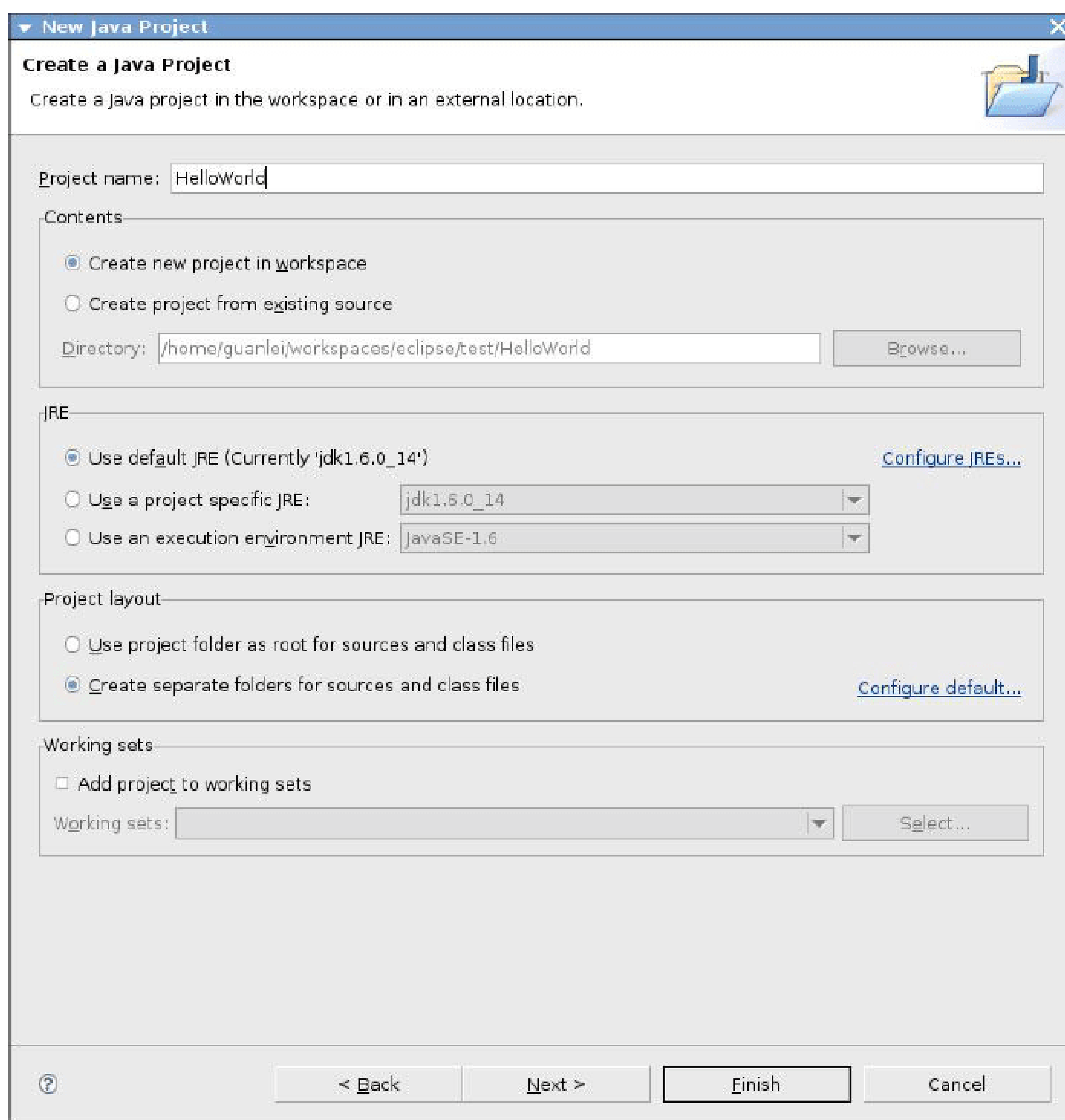


图 10.33　Linux 下的 Eclipse 中新建一个 Java 工程

（2）在此工作台下，写入 Hello World.java 的程序代码，如图 10.34 所示。

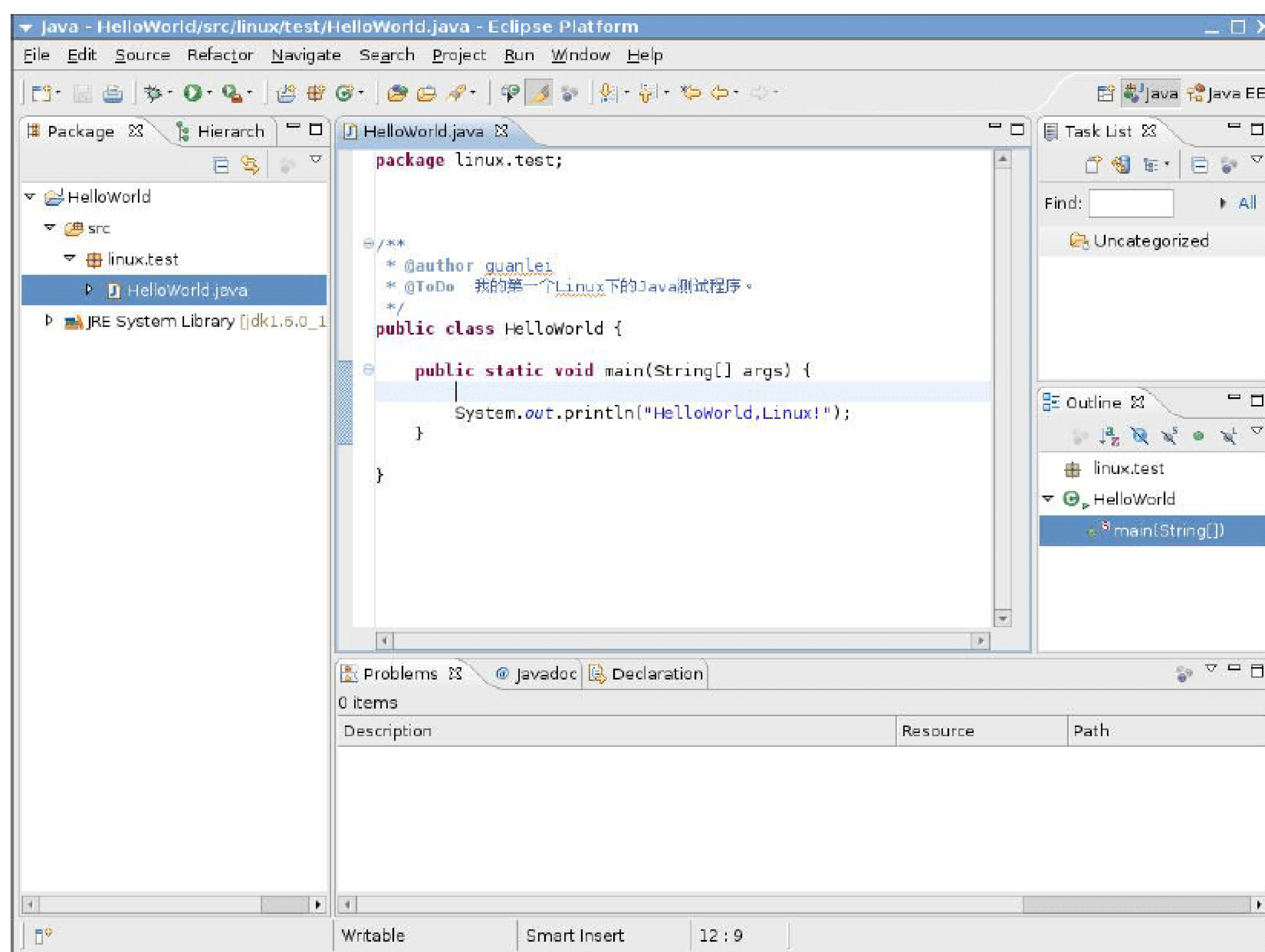


图 10.34　Linux 下编辑 Java 源代码界面

编辑完成保存后，单击源程序文件即可运行。运行结果正确且没报任何错误，说明 Linux 的 Java 开发环境安装成功，可以直接在此平台下进行 Java 的项目开发了。

10.5　本章小结

作为 P2P 应用开发的前导知识，本章重点讲解了 Java 开发环境的搭建，10.1 节简单地说明了 Java 的基础知识以及 Java 与 P2P 的渊源；10.2 节讲解了 Windows 下 JDK 的安装及环境变量的配置方法；10.3 节讲解了 Java 开发工具集 Eclipse 的安装和使用方法，最后则介绍了 Linux 平台下 JDK 和 Eclipse 的安装方法。通过本章的讲解，任何对 Java 零基础的读者都可以在 Windows 和 Linux 平台下构建自己的 Java 开发环境。

本章的知识主要面向初次接触 Java 编程的读者，通过截图和一步步的讲解，力求让读者彻底明白 Java 平台的搭建，一站式地进入 Java 项目的开发。

对有 Java 开发基础的读者可略过此章，在本书后续有关案例开发的几个章节中，对有涉及 Java 开发平台相关知识的内容本章都已经讲解，将不再另作说明。

第 11 章　Skype 的开发包及插件开发技术

在本书的第 8 章，已经详细地讲解了基于 P2P 的 Skype 通信技术，相信读者对 Skype 的系统知识已经有了比较全面的了解。Skype 作为一款即时通信工具，具有优异的扩展性能，可以将开发的各种插件附加到 Skype 系统上以定制各种应用，或通过插件技术来控制 Skype 客户端的各种操作。当前，Skype 的插件开发是一个很热门的技术，市场需求很大。

Skype 作为 P2P 技术的经典代表，读者有必要了解一些关于 Skype 的基本开发技巧。所以，本章就重点讲 Skype 的开发工具包，以及在此工具包的基础上进行 Skype 插件开发的基本方法。

本章的主要内容如下。

- Skype 开发包的基本知识：了解 Skype 开发包的功能、分类，掌握 Skype API 的基本命令。
- Skype4Java 工具包：了解 Skype4Java 包的获取方法、基本架构、文件结构及部分源代码，能读懂 Skype4Java 的参考文档。
- Skype4Java 的快速入门：掌握用 Skype4Java 进行应用程序开发的基本步骤，掌握 Skype4Java 包的一些重要对象、类、方法等。
- 用 Java 开发 Skype 应用：会搭建 Java 开发平台、能够实现最基本的通信和命令发送的功能。
- Skype 基本应用开发：掌握 Skype 的插件原理、开发方法，会用程序来实现 Skype API 的相关命令
- Skype 开发实例：掌握 Skype 的基本功能开发，包括呼叫、会话、自动应答，呼叫转接、呼叫终止等，能开发一个基于 Skype 的综合应用实例。

11.1　Skype 开发工具包使用说明

Skype 开发包，也叫 Skype 的 API 工具集，或 Skype API 类库，它提供了一套第三方的关于 Skype 的结构、应用及策略的开发方法。通过此开发包开发的应用程序，可以控制 Skype 的界面功能、增加 Skype 的额外方法和改进 Skype 的一些特性。本节主要讲解 Skype 开发工具包的基本知识。

假设你已经有了一定的使用 API 进行编程的基础，那么讲完本节的知识，就可以利用 Skype API 进行与 Skype 插件有关的编程开发了。

注意： API（Application Programming Interface，应用程序编程接口）是一些预先定义的函数，目的是提供应用程序与开发人员基于某软件或硬件的以访问一组例程的能

力，而又无须访问源码，或理解内部工作机制的细节。API 除了有应用“应用程序接口”的意思外，还特指 API 的说明文档，也称为帮助文档。

11.1.1　Skype 开发包的两个层次

Skype 的 API 有两个大的模块，通常也说有两个基本的应用层次，一个是通信层，另一个命令协议层。

通信层，主要是利用 Skype API 所提供的一系列的方法和接口所编写的应用程序，可以建立与 Skype 客户端之间的连接，并与之通信。

另一层就是命令协议层，在这一层中，Skype API 提供了一系列基于文本的“语言”命令，这些语言是用来向 Skype 客户端发送命令的。一旦应用程序与 Skype 客户端建立了连接，命令协议层与 Skype 客户端之间也就建立了一个交流的通道，应用程序就可以利用这个通道，向 Skype 客户端发送命令，通过命令来实现对 Skype 客户端的各种操作。

1. 命令层

Skype 的公共 API，使用纯文本命令协议来实现 Skype 系统与第三方应用程序之间的交互。一旦第三方应用程序连接到 Skype 上以后，它就可以向 Skype 发送纯文本的命令，同时也接收纯文格式的反馈信息。

这种机制，使用得 Skype API 成为一种或多或少与平台无关的编程语言。一旦你建立了与 Skype 进行通信的基础机制（关于 Skype 与应用程序之间通信的建立，下文会讲到），你就可以在任何你最熟悉的编程语言上使用 Skype 的 API 了，不论使用何种编程语言，用到的命令协议都是一样的。

Skype API 中的命令协议，是一种跨平台的统一格式的命令语言，不管是在 Windows、Linux 还是在 Mac 上，其格式、功能都是一样的。但是，在实践中的一些新特性、新命令，往往会首先出现在 Windows 版本中。

注意：协议层的命令语言，作为发送给 Skype 的命令，必须是基于 UTF-8 的编码格式的字符。

2. 通信层

Skype API 的语法、命令通过通信层发送给 Skype 的客户端，并通过这个层接收 Skype 客户端的反馈信息，这个通信层的实现是与平台有关的。在 Windows 系统上，使用了一种关于 wm_copydata 消息的机制，而在 Linux 系统中，它是基于 D-BUS or X11 的消息系统来完成。在不同平台上的 Skype 通信实现方法，可以参阅相应的 API 文档。

- Windows 平台下的 Skype 通信实现方法，请参考以下文档。
  https：//developer.skype.com/Docs/ApiDoc/Skype_API_on_Windows。
- Linux 平台下的 Skype 通信实现方法，请参考文档：
  https：//developer.skype.com/Docs/ApiDoc/Skype_API_on_Linux。
- Mac 平台下的 Skype 通信实现方法，请参考文档：

https：//developer.skype.com/Docs/ApiDoc/Skype_API_on_Mac。

注意：不同平台上的 Skype 开发包有很多不同的地方，但实现的功能都大体相同，想了解进一步信息的读者可到 https://developer.skype.com 的网站上查阅。

一旦实现了一种通信策略，使用应用程序与 Skype 客户端之间建立通信以后，就可以忽略一些它的实现细节，通过协议层的命令来处理应用程序与 Skype 客户端之间的会话。

总结前面对 Skype API 的说明，大概意思总结起来是这样的，当 Skype 客户端启动以后，要想控制这个客户端，需要做以下两件事情。

（1）第一件事情是要建立一个和 Skype 之间连接的通道。

Skype 插件是附加在 Skype 客户端的一种应用程序，这个应用程序要实现与 Skype 客户端之间的通信，就需要一个通信的信道，有了这个通道，作为插件的应用程序才可以向 Skype 发送各种指令。这条通道根据系统平台的不同有不同的实现机制，也就是说在 Linux、Windows 或是 Mac 中，与 Skype 的通信机制是不一样的。

而对于这条通道来说，在同一个系统平台下也可以有不同的实现机制，就好比一条路，建路的材料可以有多种，可以建成水泥路、也可以建成土路、铁路等。那么在同一平台系统上，Skype 插件与 Skype 客户端之间的通信信道，也可以由不同的语言来实现，Java、Python 等都可以。所以，Skype API 中，针对不同的语言就对应着不同的程序开发包，有 Java 的也有 Python 的，还有其他的。通道建立好了以后，通信层的工作也就完成了。接着就要做第二件事情了。

（2）第二件事情是利用这个通道，向 Skype 发送各种命令并接收这个命令的反馈信息。

向通道中发送的命令像一种“语言”一样，可以告诉 Skype 做什么、怎么做，这种命令的格式、规范、编码形式都是固定的。这样，一旦程序与 Skype 之间通过一种“语言”可能实现交流，那么就可以通过命令的形式来操作 Skype 了，这就是所说的命令层的意思。

11.1.2　针对不同编程语言的 Skype API 开发包

上节已经说过，Skype API 有多个针对不同编程语言的开发包，通过这些开发包，在实际的应用开发中，不必再在你的应用程序中建立一个到 Skype 客户端之间通信层。因为，这些开发包通过一些预编译的库文件，将这些命令协议都封装好了，作为接口提供给开发者使用。这样，就可以省去很多细节，将更多的精力放在业务逻辑和应用开发上。

当前有 3 种官方支持的 Skype API 开发工具包可供用户选择使用，这 3 种工具包分别如下。

- Skype4COM：基于 ActiveX 对象实现的 SkypeAPI，只能在 Windows 上使用。
- Skype4Py：基于 Python 语言的开发工具包，适用于 Windows、Linux 和 Mac x 系统。
- Skype4Java：Java 平台下的开发工具包，具有 Java 的跨平台性。

1．Skype4COM

Skype4COM，是当前最常用的 Skype API 库，此包的动态链接库（DDL）已经包含在了 Skype 的 Windows 的安装包里（连同插件管理器一起安装）。Skype4COM 是基于 ActiveX

对象来实现 Skype API 命令的，所以，它可以适用于任何可以访问 ActiveX 对象的程序语言中，但它不足的地方是只能在 Windows 平台上使用。

Skype4COM 的参考指南可以在网站 https://developer.skype.com/Docs/Skype4COM 上找到。

2．Skype4Py

不像 Skype4COM，Skype4Py 是针对特定的 Python 语言的库文件，也就是说，这个开发包只能使用 Python 语言进行开发。由于 Python 是跨平台的兼容性语言，所以基于 Python 的开发包不仅可以在 Windows 平台下运行，也可以在 Linux 下运行，对 Mac 系统的支持也正在研发中。

注意： 笔者在写这篇文章的时候，Skype4Py 对 Mac 系统的支持还处于研发阶段，应该很快就会有相应的版本公布。

Skype4Py 的参考指南可以在网站 http://skype4py.sourceforge.net/doc/html/上找到。

3．Skype4Java

Skype4Java 与 Skype4Py 类似，也是针对选定语言的开发工具包，它所支持的开发语言为 Java，Java 语言的特点也支持跨平台运行。后文所讲的所有实例，均是以 Skype4Java 工具包为基础进行开发的。至于 Skype4COM 和 Skype4Py 也会有所涉及。

Skype4Java 的参考指南可以在网站 http://cvs.sourceforge.jp/cgi-bin/viewcvs.cgi/skype// 上找到。

针对这几种开发包，读者可以根据自己所使用的编程语言自行选择，本文后文所讲的示例都是基于 Skype4Java 开发包完成的。

11.1.3　Skype API 的命令

上文已经说过，要实现对 Skype 的操作，还需要向 Skype 发送各种命令。要学习 Skype API 的命令协议，建议首先下载一个 Skype API 的 Tracer 程序，这个小软件是一个简易的小工具，利用它可以测试各种 Skype API 的命令。它的下载地址是

https://developer.skype.com/Download?action=AttachFile&do=get&target=Tracer.exe；

API Tracer 运行起来后，运行界面如图 11.1 所示。

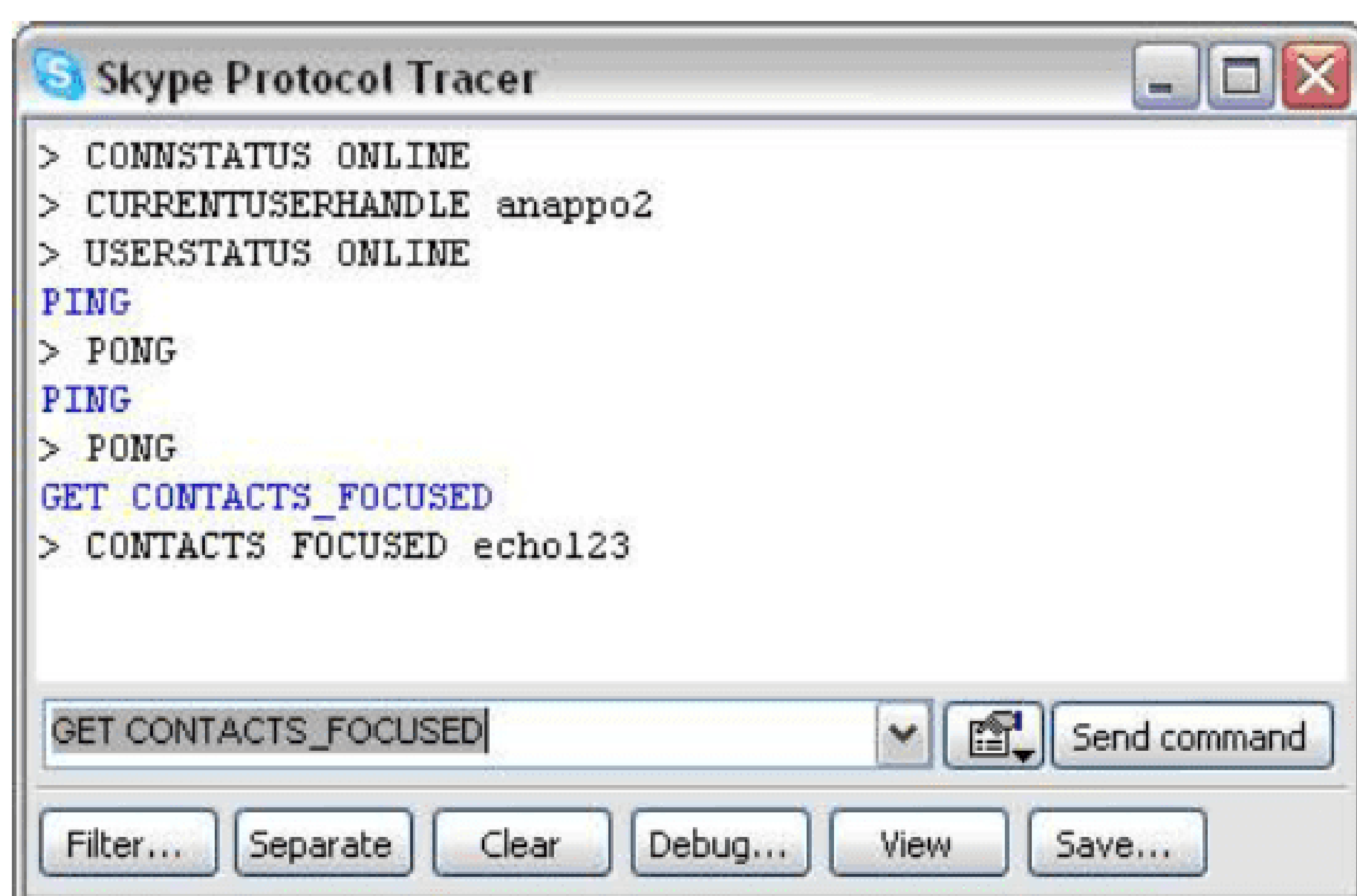


图 11.1　Skype Tracer 运行界面图

它类似于一个“终端”界面，通过这个界面，你可以向 Skype 发送各种命令，并可以从这个界面中即时看到针对这条命令而来自 Skype 的反馈信息。

注意： 在后文的实例中，会带领读者一起开发一个这样的 Tracer 小程序。

从图 11.1 中可以看到，发送给 Skype 的命令有 PING、GET CONTACTS_FOCUSED 等，这些命令必须符合 Skype API 中规定的命令形式和格式的要求。可以在 Skype 公用 API 参考指南上阅读到关于 Skype 命令协议的详细要求。

Skype 的文档信息是进行 Skype API 开发不可缺少的工具，其下载地址是 https://developer.skype.com/Docs/ApiDoc。

注意： 如果只是查阅 Skype API 的命令的语法格式、使用方法、参数要求等，可直接进入网站 https://developer.skype.com/Docs/ApiDoc/Commands，进行查看。

1．Skype API 的命令格式

在 Skype API 中基本的命令格式如下。

- 命令的标识：command_id；命令标识符，对识别与某一特定命令的响应有很重要的作用，它支持 Skype API 中大多数的命令，也同样适用于来自 Skype 的应答命令。
- 语法：#<command_id>command；
- 应答：#<command_id>response|error；
- 参数：command_id - client assigned alphanumeric identifier（客户端指定的字母或数字的标识）；
- 错误：all possible errors for a given command（所有的，Skype API 命令中可能出现的错误信息）；
- 版本：Protocol 4。

2．Skype API 命令使用方法

- 一个简单的应答命令。

```
-> #AB GET USERSTATUS
<- #AB USERSTATUS ONLINE
```

在这一对命令中，->#AB GET USERSTATUS 命令是发向 Skype 的，<-#AB USERSTATUS ONLINE 命令是来自 Skype 的应答。这两个命令也可以由标识看出，标识“->”代表发送，而标识“<-”代表应答。

在发送命令中，AB 就是 command_id 命令的标识的意思，这个标识的作用就是给这条命令起个名字，名字可以随便取，只要符合命令标识的命名规则即可。GET USERSTATUS 是命令语句，从字面意思就可以理解，这条命令的意思从 Skype 客户端取得用户的状态信息。

在应答命令中，AB 也是命令的标识，证明 Skype 回答的是名字叫 AB 的命令，USERSTATUS ONLINE 是应答信息，表示当前用户在线。

还有其他的一些命令示例，对照着命令格式的说明，很直观地就能看出这些命令的意思。

- 一个无效的命令将返回一个错误信息。

```
-> #123 GET XZY
<- #123 ERROR 7 GET:  invalid WHAT
```

因为这个命令根本不存在，所以返回错误信息 ERROR。

Skype 除了应答用户的命令信息外，还会针对用户发送的命令发出一些通知消息，如下就是命令的响应和通知示例。

- 命令响应和发送通知消息：

用户发送一个命令，将当前用户设为在线状态。

```
-> #cmd11 SET USERSTATUS ONLINE
```

这个命令是来自 Skype 的应答。

```
<- #cmd11 USERSTATUS ONLINE
```

下面这个命令就是一个通知，这个通知消息会在处理完用户的请求之后发送出去。

```
 <- USERSTATUS ONLINE
```

以上过程可用如下直观的语言来描述。

（1）用户（请求）：把当前用户设置为在线状态。

（2）Skype（回答）：我已经设置成在线状态了。

（3）Skype（通知）：通知你一下，当前状态是在线的。

Skype 还可以进行异步的命令响应和通知，也可以在响应命令之间出现通知消息，如下是两个命令的示例。

- 异步的命令响应和通知。

```
-> #50 CALL +18005551234
```

以下这个事件，是在应答消息到来之前发生。

```
<- CALL 651 STATUS ROUTING
<- #50 CALL 651 STATUS ROUTING
<- CALL 651 PSTN_STATUS 10503 Service Unavailable
```

以下的事件并没有命令标识。

```
<- CALL 651 FAILUREREASON 1
<- CALL 651 STATUS FAILED
```

- 通知消息可以出现在命令与响应之间。

```
-> #50 PING
```

其他的事件可以在应答命令回复之前到达。

```
<- USER echo123 LASTONLINETIMESTAMP 1105764678
<- USER echo123 FULLNAME Echo Test Service
<- USER test LASTONLINETIMESTAMP 1105487965
```

以下才是来自 Skype 的就任命令。

```
<- #50 PONG
```

以上通过几个简单的示例，说明了一下对 Skype API 命令格式的理解，下面是一些 Skype 开发中常用的几种命令操作，读者有必要先了解一下，后文的程序实现会用到相关知识。

3．Skype API 命令举例

- 可以通过如下这种方式来查询联系人记录（用户对象）的各种属性信息。

```
-> get user echo123 birthday
<- USER echo123 BIRTHDAY 0
-> get user echo123 is video capable
<- USER echo123 IS_VIDEO_CAPABLE FALSE
```

- 可以用如下的这种命令方式来测试呼叫 Skype 的通话测试服务。

```
-> call echo123
<- CALL 14662 STATUS UNPLACED
<- CALL 14662 STATUS UNPLACED
<- CALL 14662 STATUS ROUTING
<- USER echo123 COUNTRY United Kingdom
<- USER echo123 COUNTRY United Kingdom
<- USER echo123 COUNTRY
<- CALL 14662 STATUS RINGING
<- USER echo123 COUNTRY United Kingdom
<- CALL 14662 VAA INPUT STATUS FALSE
<- CALL 14662 STATUS INPROGRESS
<- CALL 14662 DURATION 1
<- CALL 14662 DURATION 2
<- CALL 14662 DURATION 3
<- CALL 14662 STATUS FINISHED
```

- 使用如下命令可以创建一个会话。

同一个目的对象创建一个会话。

```
-> CHAT CREATE anappo5
<- CHAT #anappo/$anappo5;2e4e763a2fc121ed STATUS DIALOG
-> OPEN CHAT #anappo/$anappo5;2e4e763a2fc121ed
<- OPEN CHAT #anappo/$anappo5;2e4e763a2fc121ed
```

直接创建会话。

```
-> CHAT CREATE
<- CHAT #anappo/$72cb4c9d0871e6dc NAME #anappo/$72cb4c9d0871e6dc
<- CHAT #anappo/$72cb4c9d0871e6dc ACTIVITY_TIMESTAMP 0
<- CHAT #anappo/$72cb4c9d0871e6dc STATUS MULTI_SUBSCRIBED
<- CHAT #anappo/$72cb4c9d0871e6dc TYPE MULTICHAT
<- CHAT #anappo/$72cb4c9d0871e6dc STATUS UNSUBSCRIBED
<- CHATMEMBER 570 ROLE USER
<- CHAT #anappo/$72cb4c9d0871e6dc MYROLE USER
<- CHAT #anappo/$72cb4c9d0871e6dc MEMBERS anappo
<- CHAT #anappo/$72cb4c9d0871e6dc ACTIVEMEMBERS anappo
<- CHAT #anappo/$72cb4c9d0871e6dc MYSTATUS SUBSCRIBED
<- CHAT #anappo/$72cb4c9d0871e6dc STATUS MULTI SUBSCRIBED
<- CHAT #anappo/$72cb4c9d0871e6dc TIMESTAMP 1175089677
-> OPEN CHAT #anappo/$72cb4c9d0871e6dc
<- OPEN CHAT #anappo/$72cb4c9d0871e6dc
```

同时与两个目的对象创建会话。

```
-> CHAT CREATE anappo3， anappo5
<- CHAT #anappo/$8c9e3bb94643d668 NAME #anappo/$8c9e3bb94643d668
<- CHAT #anappo/$8c9e3bb94643d668 ACTIVITY_TIMESTAMP 0
<- CHAT #anappo/$8c9e3bb94643d668 STATUS MULTI_SUBSCRIBED
<- CHAT #anappo/$8c9e3bb94643d668 TYPE MULTICHAT
<- CHAT #anappo/$8c9e3bb94643d668 STATUS UNSUBSCRIBED
<- CHATMEMBER 585 ROLE USER
<- CHAT #anappo/$8c9e3bb94643d668 MYROLE USER
<- CHAT #anappo/$8c9e3bb94643d668 MEMBERS anappo
<- CHAT #anappo/$8c9e3bb94643d668 ACTIVEMEMBERS anappo
<- CHAT #anappo/$8c9e3bb94643d668 MYSTATUS SUBSCRIBED
<- CHAT #anappo/$8c9e3bb94643d668 STATUS MULTI SUBSCRIBED
<- CHAT #anappo/$8c9e3bb94643d668 TIMESTAMP 1175089858
<- CHAT #anappo/$8c9e3bb94643d668 MEMBERS anappo anappo3 anappo5
<- CHAT #anappo/$8c9e3bb94643d668 FRIENDLYNAME anappo3， anappo5
-> OPEN CHAT #anappo/$8c9e3bb94643d668
<- OPEN CHAT #anappo/$8c9e3bb94643d668
```

- 使用如下命令，在 Skype 客户端创建一个菜单项。

```
-> CREATE MENU_ITEM test01 CONTEXT contact CAPTION "TEST 01" ENABLED true
<- MENU_ITEM test01 CREATED
```

以下的菜单项只在 SkypeOut 模式进行连接时才会起作用。

```
Following menu item will only be enabled for SkypeOut contacts
-> CREATE MENU ITEM test02 CONTEXT contact CAPTION "TEST FOR SKYPEOUT"
CONTACT TYPE FILTER skypeout
<- MENU_ITEM test02 CREATED
```

注意： 要深入学习 Skype API 知识或进行较大型的 Skype 开发，Skype API 中的命令是非常重要的。详细知识请参考 https://developer.skype.com/Docs/ApiDoc/Commands 上的文档说明。

在 11.2 节里，就针对具体的 Skype API 的开发包进行讲解，详细地阐述一下 Skype4Java 包的基本架构和核心的实现代码。

11.2　Skype4Java 的工具包及目录结构

Skype4Java，意思就是 Skype for java，是一种用 Java 语言实现的 Skype 工具包，由于 Java 语言的跨平台性，所以这个工具包在任何系统下都可以使用。至于上文讲到的另外两种 Skype 开发包的实现方案 Skype4COM、Skype4Py 等，读者可以自己对照着去研究。

11.2.1　Skype4Java 简介及基本构架

Skype4Java，简单地说就是一组类文件、一个 Java 库，就是利用 Java 语言的特性开发出来的 Skype 外壳应用程序的一组类库。利用 Skype4Java 的工具包，可以通过 eclipse、IntelliJ 和 Net Beans 等开发工具开发出跨平台的、基于 Java 的各种 Skype 应用插件。

Skype4Java 工程，是一个免费开源的 Java 工程，由 Koji Hisano 为领导的开发小组下

负责维护和更新。2006 年 9 月 30 日发布了 Skype4Java 1.0 版，当前 Skype4Java 的最新版本可以在网站 https://developer.skype.com/wiki/Java_API 上下载得到，也可以到 http://sourceforge.jp/cvs/view/skype/网站上，通过 CVS 工具，将工程源代码 Check out 到本地。

注意： CVS（Concurrent Version System）版本控制系统，主要用于在多人开发环境下的源码的维护和版本的控制。CVS 对于个人开发有很大用处，是大型软件开发尤其是分布团队开发必不可少的工具。

Skype4Java 提供了一个较为完善的开发体系，使用户可以非常容易地使用 Java 开发 Skype 外壳程序。Skype4Java 的实现是按照分层的方法来设计的，分层架构如图 11.2 所示。

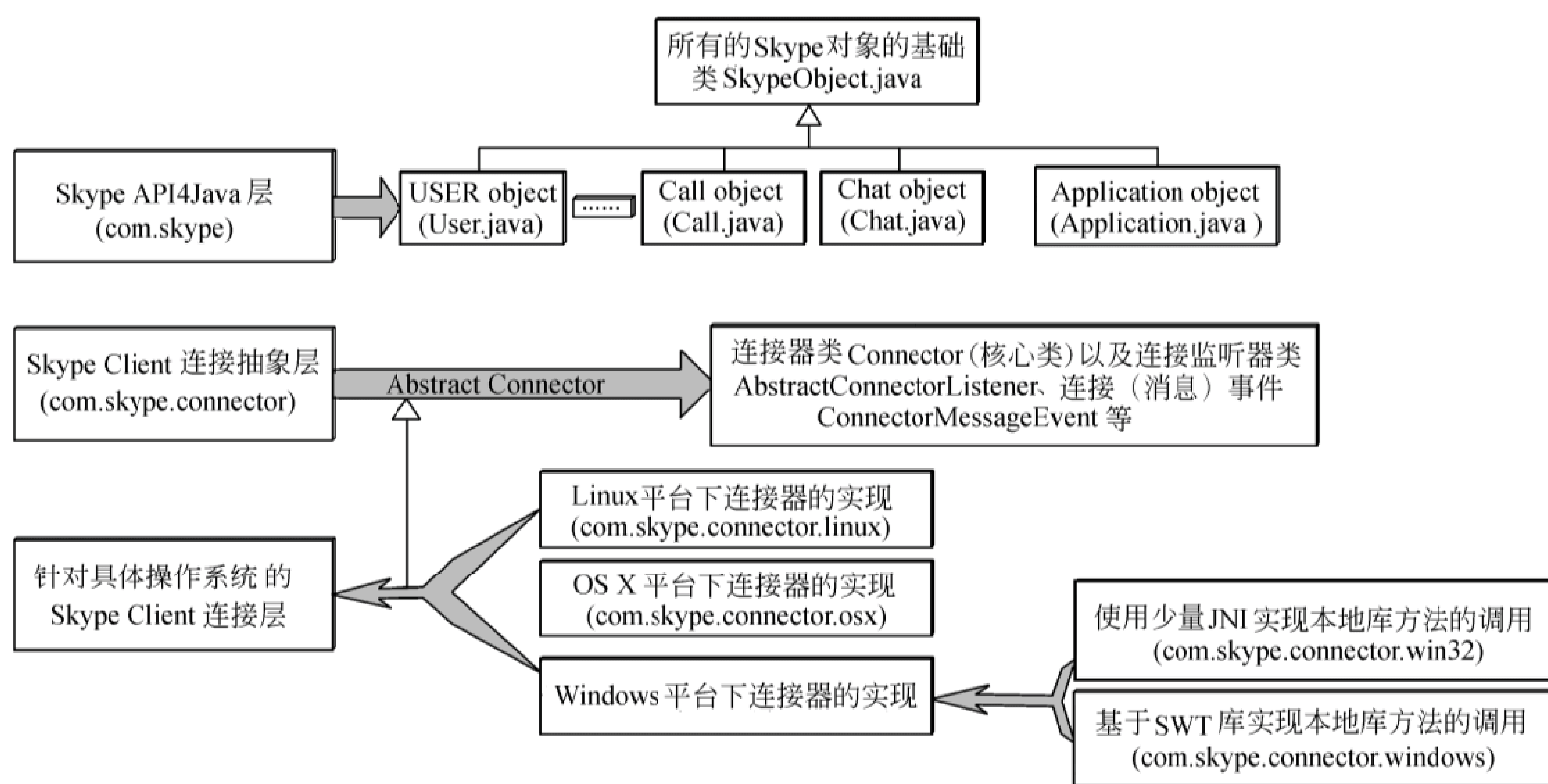


图 11.2　Skype4Java 的分层结构图

Skype 插件程序与 Skype 的通信都是基于一些文本命令来完成的。因此，要想开发 Skype 的插件应用，就需要依靠不同操作系统平台的系统调用，发送 Skype 平台的相应命令来实现。

Skype4Java 在最底层，也是通过操作系统的系统调用来完成消息的发送。针对异构平台，Skype4Java 提供了不同平台的底层实现，而 Skype4Java 的使用者不必考虑具体的操作系统平台。因为抽象父类 Connector 类的静态方法 getInstance()具体判断当前操作系统是什么平台，采用简单的工厂模式，返回相应平台的子类对象，如 OSXConnector.、LinuxConnector、WindowsConnector、Win32Connector 等。

以上这些针对不同平台的 Connector 类都是 Connector 类的子类。Connector 及其子类实现了一些系统调用的方法，其中被重载的一系列 execute()方法最终调用不同子类的 sendCommand (String command)方法，实现了不同平台的消息发送。

而对于一些数据的接收和处理全局都是采用事件监听的机制（观察者模式），用于完成数据的接收、处理机制的调用、错误的处理等。

Skype 平台利用错误消息机制提供了出错的处理。因此，在处理错误的过程中，只需分析错误消息的内容，然后通过观察者调用相应的处理机制即可。

11.2.2　Skype4Java 源代码包的文件结构

Skype4Java 的开发包可以下载 Jar 形式的打包文件，也可以下载源文件，开发的时候可以直接将 Skype 包导入到构建路径下，也可以将源代码导入到自己的工程文件里。本文为代码讲解需要，直接将其源文件附加到要开发的 Java 工程里。

读者可以在网站 http://sourceforge.jp/cvs/view/ skype/ 上找到 Skype4Java 的工程源代码，用 CVS 工具可以将其 Check out 到本地。

也可以在 Skype 的开发网站上，直接下载 Skype 的源码包。当前的版本是 Skype_1.0 版。把 Skype 的源码包下载到本地后，它的文件结构如图 11.3 所示。

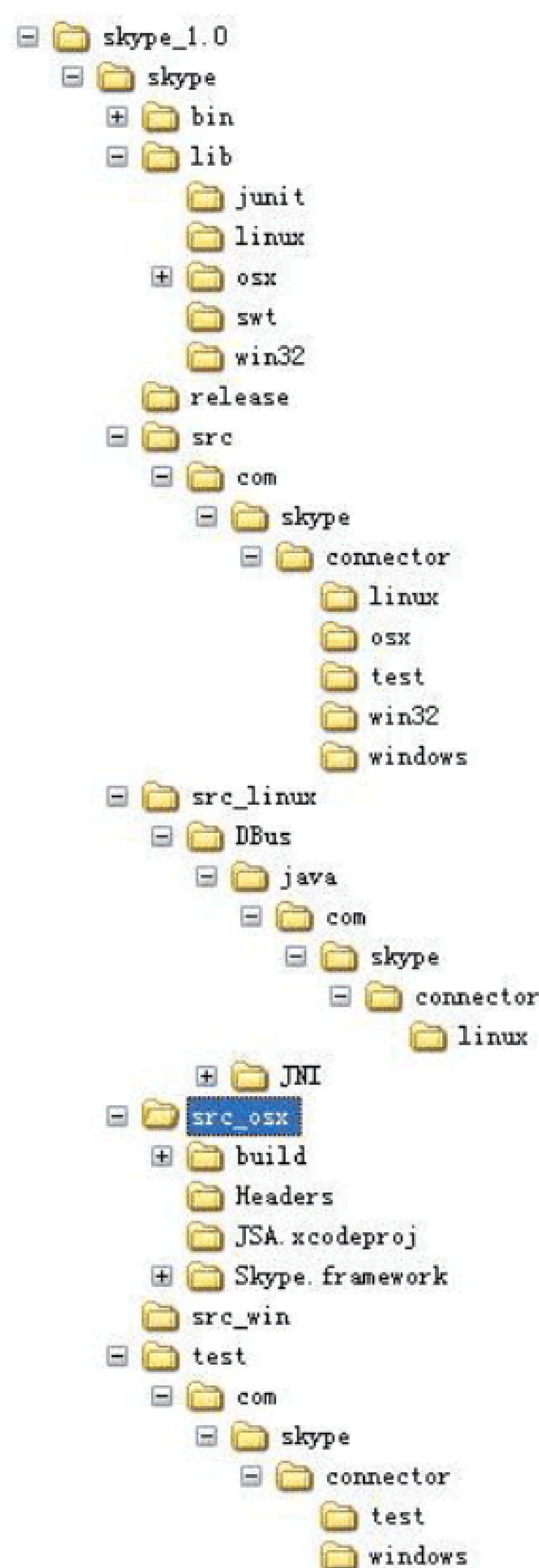


图 11.3　Skype 源码包的目录结构

从图中可以清楚地看出 Skype 的源码包的目录结构及层次关系。下面简要说明这些目录的作用。

- Bin：用于存放 Skype 的类文件，即 class 文件。
- Lib：Skype 开发中用到的主要类库，需要导入到工程中的一些 Jar 包都可以在这里找到。
- Release：都是一些可用于发布的 Jar 包文件。
- Src：Skype4Java 包中源代码所在的文件夹。
- Src_linux、src_osx、src_win：这 3 个目录，主要是针对不同平台的一些底层的实现通信连接的方法。
- Test：主要用于存放一些测试文件的目录。

注意：以上这些目录下文件的详细内容，读者可自行查看，所有的源代码也可以在随书光盘中找到。

11.3　Skype4Java 的入门和快速开发

使用 Skype4Java 的库进行相关开发，需要熟悉 Skype 的相关知识，所以，在学习本节之前，建立读者认真阅读本章前几节的关于 Skype 的知识点，并熟练掌握 Skype 基本使用和操作方法。

Skype4Java 的 API 工具包，提供了很多 Skype 客户端的基本功能，如访问联系人列表，拨打和接听电话，进行电话会议，以及不同应用程序状态、消息的传递和进行聊天等。在讲解具体的应用开发之前，首先介绍一下如何入门 Skype4Java，如何利用 Skype4Java 进行快速开发。

11.3.1　Skype4Java 的基本开发步骤

相信本书的读者，很多都是初学 P2P，也是初次接触 Skype4Java 的开发包。当面对一个陌生的软件工具包的时候，如何利用这个包进行编程开发呢，下面简要说一下基本的开发步骤。

（1）得到这个软件包及其支持文档：Skype4Java 软件包可以从网站上免费得到，也可以同时得到相应的参考文档，这些在 10.2 节中已经做过了。

（2）根据参考文档来阅读核心代码：在参考文档中，大部分对一些核心的关键代码都有详细的说明，那么就以此为基础去读这些核心的源代码。这些源代码中一般都有注释，看看这些源代码中的类是如何实现的，方法如何调用，核心功能体现在哪个方法上等。

（3）阅读示例代码：如果有的话阅读一些示例代码，看看这些示例如何调用开发包中的方法，如果开发包中没有提供，可以在互联网上搜索一下，看看别人是怎么做的。

（4）编写测试程序：明白了一些基本的调用方法，就可以编写一些测试程序进行测试，也许只有一两行代码，但能实现一个小功能。

（5）过程调试：调试是编程过程中必不可少的一个环节，尤其在学习一些新知识的时候。在程序中设置一个断点，然后跟着这个断点一步步执行下去，就能明白程序的整个执行流程。多调试几次、多问几个为什么，理解会更深。

（6）重要方法和对象总结：做完了以上几个过程，相信读者对整个程序的架构、重要的方法对象已有了比较直观的了解，那么就可以对工具包中一些重要的对象和方法进行总结。理解它们是如何调用的、需要传入什么参数、返回什么对象、可以实现什么功能、会不会抛出异常、依赖哪些包和类等，总结了这些以后，对后面的开发非常用

（7）进行测试开发：做完上面这些，就可以试着开发自己的小程序了。

当然，用 Skype4Java 进行开发，还有很重要的一步，就是下载并安装 Skype 客户端，这是必须的，因为 Skype4Java 工具包所开发的应用程序，需要借助 Skype 客户端来实现，所以，必须要有一个 Skype 客户端才能保证程序正常地运行。

在以上的这几个过程中，已经分析了 Skype4Java 的架构和核心代码，下面就总结一下 Skype4Java 中的重要对象和调用方法。

11.3.2　Skype4Java 的重要对象之——呼叫（Calls）

呼叫（Calls），就是通过应用程序向 Skype 客户端发送命令，以呼叫好友列表中的用户。它是 Skype 应用开发中一个非常重要的应用，用来进行与呼叫有关的各种操作，这些操作都被封装在一个叫 Call.java 的类里。

Call.java 类，在 Skype 源代码包中，全路径为 com.skype.Call.java。Call 类的全局视图如图 11.4 所示。

图 11.4 中只展示了 Call 类一些主要的方法，详细的方法体及内容请参阅源代码。

通过 Call.java 类，要产生一个呼叫，只需使用如下一个调用命令，需要传入一个 SkypeID 的参数，参数内容就是好友列表中用户的 ID。

```
Skype.call (String SkypeID)
```

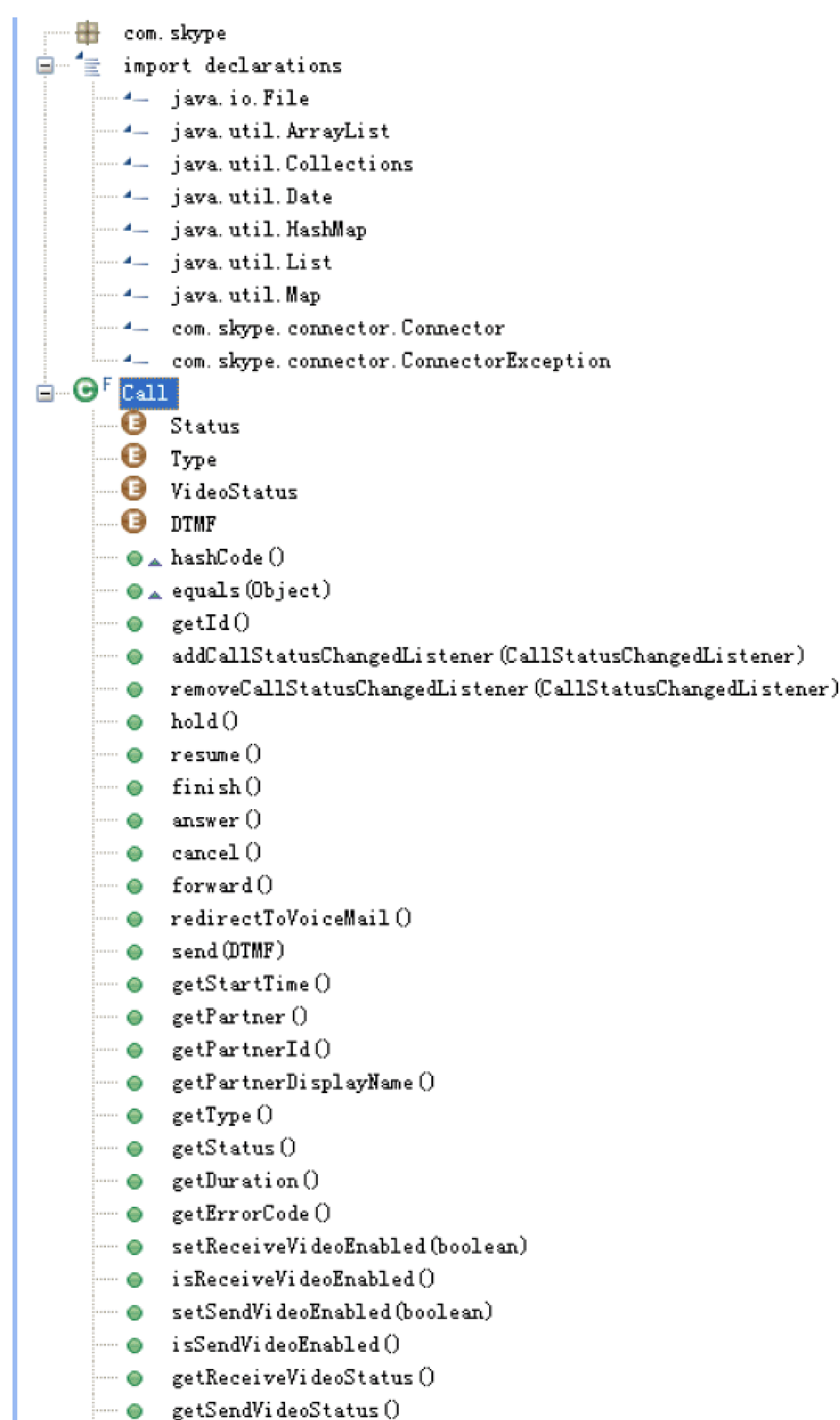


图 11.4　Call.java 类的全局视图

调用这个方法以后，它将返回一个 Call 对象。如果参数是一个 Skype ID 的列表，也就是呼叫多个用户，那么将返回一组呼叫（电话会议的形式），得到 Call 对象以后，就可以对各种呼叫操作进行管理。要想终止呼叫，只需调用它的 Finish()方法即可。示例代码如下：

```
Call myCall = Skype.call("ta-4170");    //创建一个 Call 对象
myCall.finish();                        //终止一个呼叫
```

当要接收并应用一个呼叫时，可以通过 Skype.addCallListener 的操作，来安装一个呼叫监听器（CallListener）。这个 CallListener 要实现相应的 callReceived()和 callMaked()的方法，这些方法在只在接收到呼叫或产生一个新的呼叫时才起作用。一旦处于被叫的地位，可以用一个 call.answer()的方法来应答这个呼叫。

创建一个呼叫监听器的方法如下：

```
Skype.addCallListener(myCallListener);    //增加一个呼叫监听器
```

Skype4Java 利用 Call.java 类还可以实现一个重要的应用，就是创建一个组会话，其方法就是在现有的呼叫的基础上，再加入一个新用户以创建一个 Conference Call（电话会议）。要实现这个功能，就是先创建一个 Call 对象，再在此基础上使用 Skype API 的

JOIN_CONFERENCE 命令，就可以实现创建电话会议的目的。实现的部分代码如下：

```
Call myCall = Skype.call("userA");    // 呼叫第一个好友，userA
String firstID = myCall.getId();
```

确保第一个呼叫正在处理中，需要时刻注意的是，要能正确地处理监听器所监听到的状态改变的信息，可以通过 CallStatusChangedListener 这个监听器来实现对状态改变情况的监听。

```
while (myCall.getStatus() != Call.Status.INPROGRESS) ;
myCall.hold ();
```

例如呼叫第 2 个用户，当第 1 个呼叫正在处理中的时候，可以把第 2 个呼叫加入到第 1 个呼叫中。

```
Call secondCall = Skype.call("userB");     //呼叫第二个用户，userB
String secondID = secondCall.getId ();
```

同样，在这个过程中，也需要通过呼叫状态监听器（CallStatusChangedListener）监听并处理状态改变的情况。

通过 while 循环来判断状态信息。

```
while (econdCall.getStatus() != Call.Status.INPROGRESS) ;
                                                    //对当前的状态进行判断
try {
//调用 Connector
 Connector.getInstance().executeWithId("SET CALL "+secondID+ " JOIN
CONFERENCE "+firstID, "CALL");
  } catch (ConnectorException ex) {
     ex.printStackTrace ();
   }
```

注意： 在这个示例代码中，为了方便起见，用了一个 While 循环等待的方法来进行监听。在实际应用中，这个方法是不可取的，最好用 CallStatusChangedListener 来实现监听的功能。

11.3.3　Skype4Java 的重要对象之——聊天（Chat）

Chat 是 Skype4Java 的另一个重要对象，通过 Chat 可以实现消息的发送和接收、实现对会话过程的各种操作。Chat 对象由 com.skype.Chat.java 类来封装，含有各种对 Chat 行为的操作。Chat.java 类的全局视图如图 11.5 所示。

图 11.5 所示的图中，可清楚地看出 Chat 所封装的所有方法，调用这些方法就可以实现各种 Chat 操作。

聊天与呼叫类似，如果想接收到一个消息，同样需要安装一个聊天监听器，示例代码如下：

```
Chat  myChat = Skype.chat("skype");//创建一个名为 skype 的 Chat 对象
myChat.send("Hello");                 //通过这个 Chat 对象将内容为 Hello 的消息发送
                                        出去
```

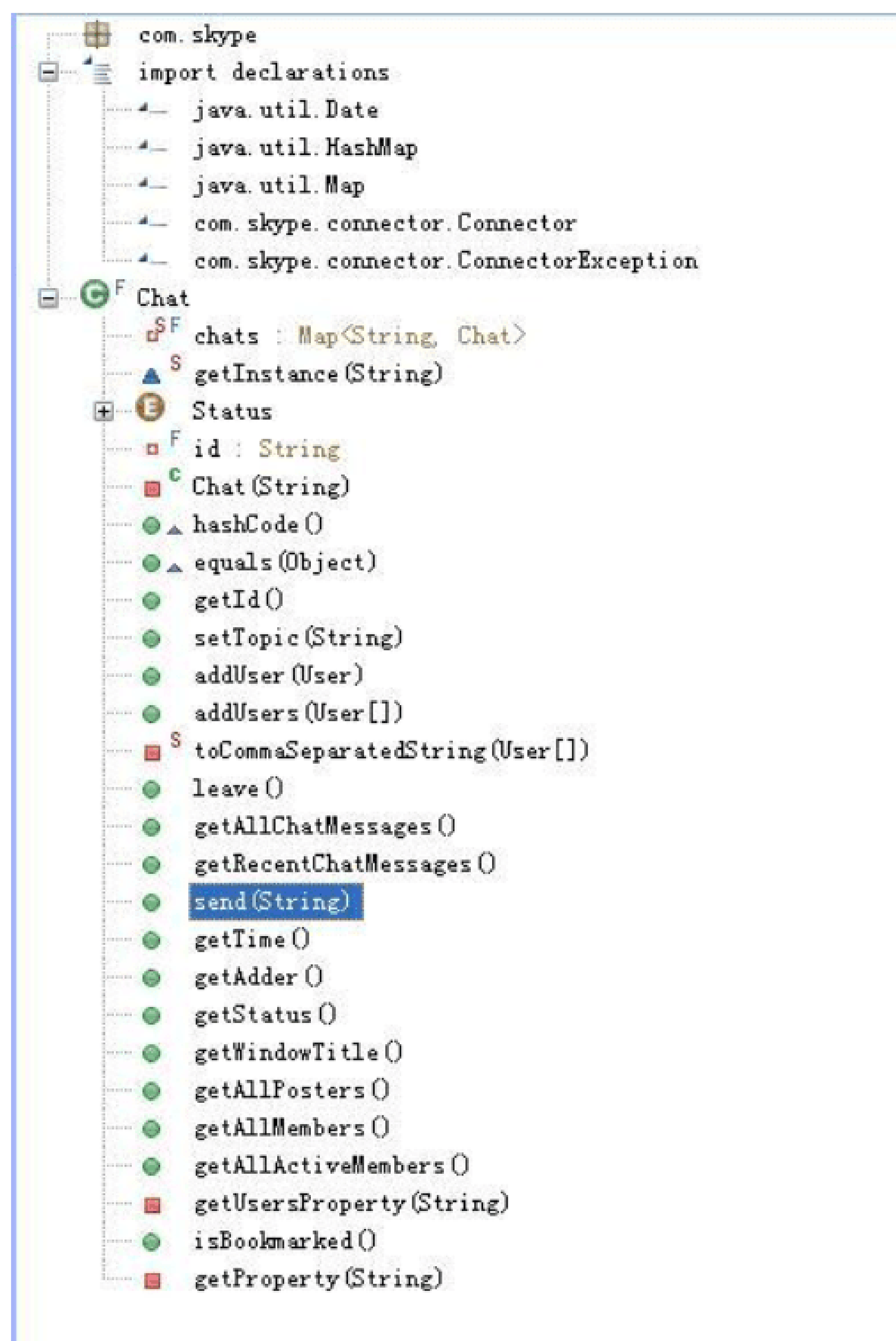


图 11.5　Chat.java 类的全局视图

11.3.4　Skype4Java 的重要对象之——连接（Connector）

Skype4Java 提供了连接层为实现各种命令的发送，这个连接过程是由 Skype 源代码包中的 Connector 类来实现的。

Connector 类在 com.skype.connector 包下，在不同的系统平台下，由不同的 Connector 实例底层实现方案。关于 Connector 的全局视图如图 11.6 所示。

图 11.6 展示了 Connector 类的主要方法，还有其他的方法请参考源代码。Connector 为类的主要作用，可以用于建立一个连接，然后发送 Skype API 的各种命令。

例如，想通过命令找出最后一次在线的用户，只需取得一个连接到 Skype 的 connector 实例，然后执行相关的命令即可，程序实现方法如下：

```
try {
String response = Connector.getInstance().executeWithId("GET USER ta-4170
LASTONLINETIMESTAMP", "USER");
System.out.println ("Response is " + response);
        } catch (ConnectorException e) {
            System.out.println (e.toString());
        }
```

注意： 在使用 executeWithId 这个方法的时候，需要相应的命令和响应的头消息，提供的响应头必须与要接收的头消息保持一致，否则将得不到任何应答。

关于 executeWithId()方法，它是 Skype4Java 包中进行应用开发时经常用到的一个方法，

主要用来执行一个 Skype API 的命令，此方法的原型如下：

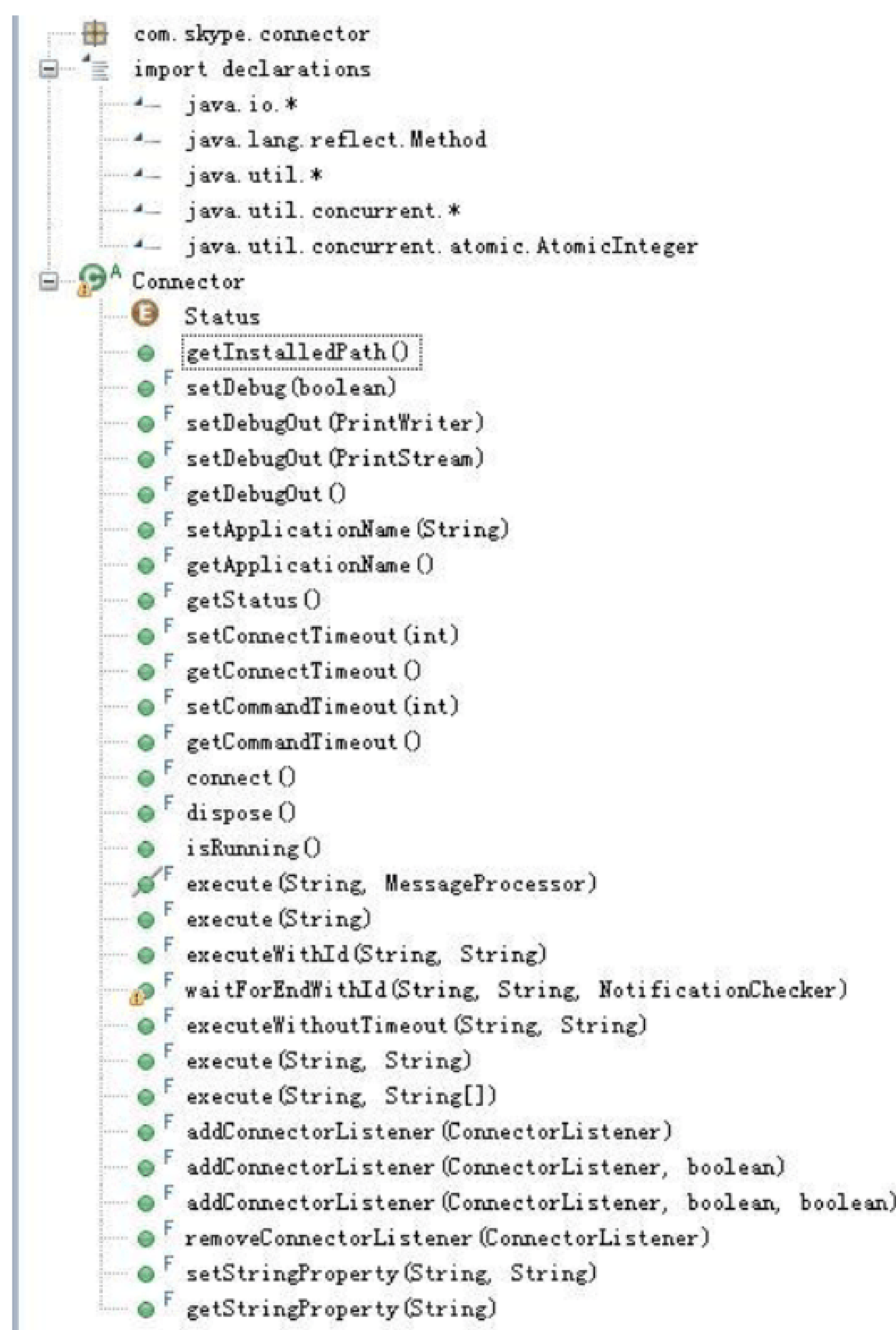


图 11.6　Connector.java 类的全局视图结构

```
public final String executeWithId(final String command, final String
responseHeader) throws ConnectorException {
        ConnectorUtils.checkNotNull ("command", command);
        ConnectorUtils.checkNotNull ("responseHeader", responseHeader);
        final String header = "#" + _commandCount.getAndIncrement() + " ";
        final String response = execute(header + command, new String[] { header
        + responseHeader, header + "ERROR " }, true);
        return response.substring(header.length());
    }
```

这个方法中，首先是对这些命令的方式和格式的判断，然后调用 execute()方法执行命令，所以，重要的实现过程就是 execute()方法。关于这个方法中有多种不同的实现，在后文会有相关的讲解。

11.3.5　Skype4Java 的简单示例

上文已经分析了 Skype4Java 中的重要对象和方法，现在就利用 Skype4Java 工具包，编写一个简单的 Java 程序示例，使其能够得到 Skype 客户端的版本信息。编程步骤与实现方案如下。

（1）创建一个新的 Java 应用程序，将下载的工具包的库文件导入到新建 Java 工程的构建路径下，并将以下的类导入到应用程序中。

```
# mport com.skype.ContactList;
# import com.skype.Friend;
# import com.skype.Skype;
# import com.skype.Application;
# import com.skype.SkypeException;
# import com.skype.Stream;
```

（2）编写应用程序，要得到 Skype 的版本信息，Skype4Java 的工具包中提供了一个 getVersion()的方法，调用这个方法可就以得到 Skype 的版本信息。调用此方法的过程如下：

```
System.out.println(Skype.getVersion());
```

（3）如果能返回并打印出正确的 Skype 版本信息，证明所有的安装、操作都是正确的。以下就是用 Skype4Java 工具包获取 Skype 客户端版本信息的编程方法，完整的编程代码如下：

```
import com.skype.Skype;
import com.skype.SkypeException;
public class Main {
    public static void main(String[] args) {
        try {
            System.out.println (Skype.getVersion ());
        } catch (SkypeException ex) {
            ex.printStackTrace ();
        }
    }
}
```

开发其他不同功能的 Skype 应用程序，其实现步骤与此类似，首先需要引入 Skype4Java 的相关类库，然后通过方法调用、接口实现、编写业务逻辑等来实现应用程序的相关功能。

注意：理解这些 API 的最好方式就是阅读 Skype API 的操作文档，以及 Skype4Java 的 Java 文档。

以上只是 Skype4Java 的基本功能和使用方法介绍，11.4 节将通过实例的方式来讲解利用 Skype4Java 开发 Skype 的基本应用。

11.4　基于 Java 平台开发 Skype 应用

Skype4Java 开发工具包，是一个在 Java 平台上开发 Skype 相关应用的类库。通过 Skype4Java 中提供的接口和方法，可以开发一些基于 Skype 客户端的插件和其他的应用程序。在此工具包的基础上，可以很容易地实现应用程序与 Skype 客户端、Skype 服务器之间的通信，可以通过协议命令的形式来对 Skype 客户端进行如呼叫、聊天、连接等操作。本节就以实例的形式讲解，如何通过 Skype4Java 提供的方法和接口来实现一系列操纵 Skype 客户端的功能。

11.4.1　Skype 开发框架的搭建

本节将从新建一个 Java 工程开始，一步步引导读者利用 Skype4Java 包，通过 Java 语

言来开发 Skype 的基本应用，首先就从 Skype 开发框架的搭建说起。

1．新建 Java 工程

打开 eclipse 开发平台，在菜单项中，选择 File 选项，在弹出的下拉菜单中选择 New|Java Project 命令。在弹出的对话框中输入工程的名字 skype_dev_eg（Skype 开发示例的意思），如图 11.7 所示，然后单击“确定”按钮。

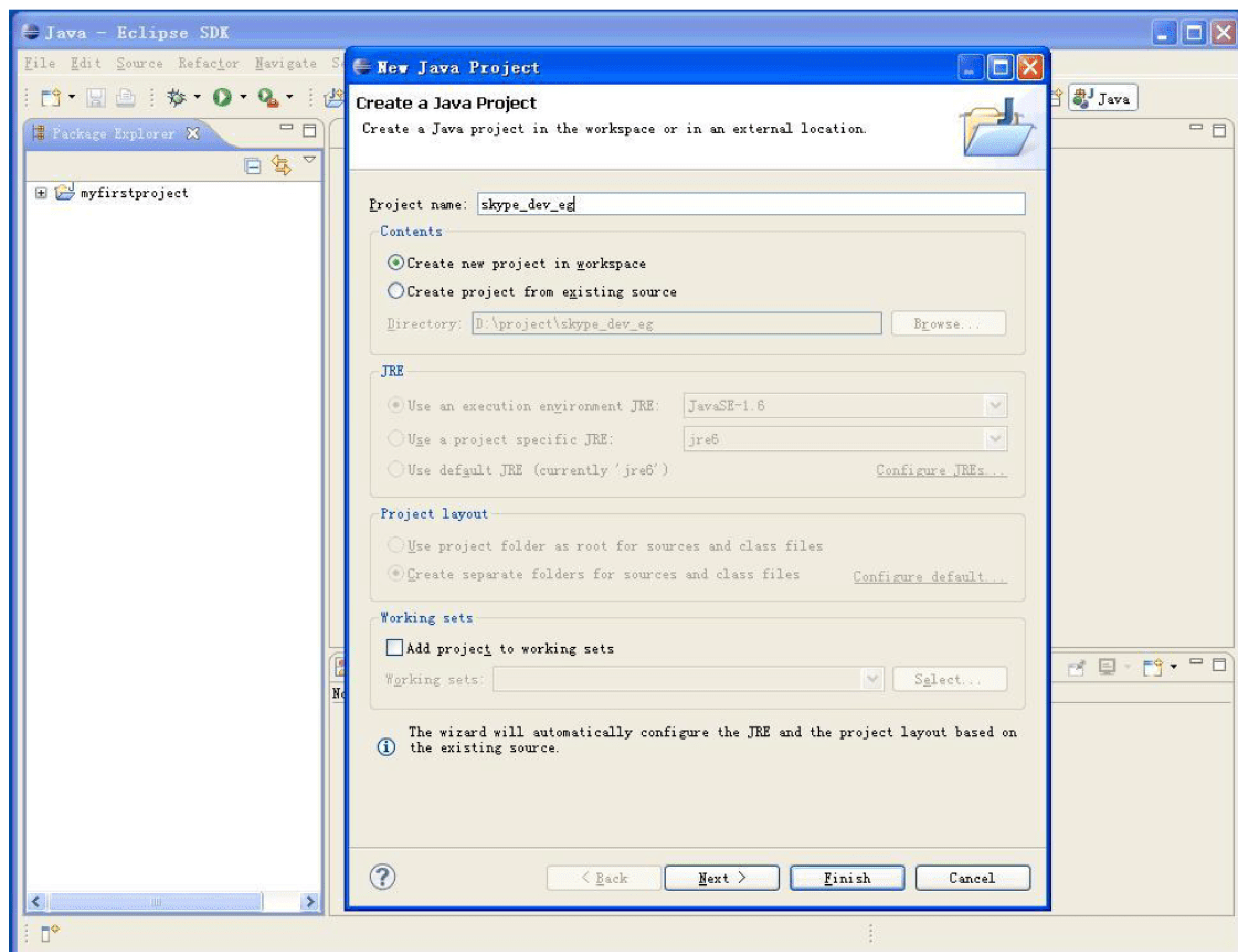


图 11.7　新建 Java 工程的界面

这样，一个名为 skype_dev_eg 的空项目就建好了，下文中所有的关于 Skype 基本开发的示例都在这个工程下进行。

2．导入 Skype4Java 的源代码

解压缩下载后的 Skype4Java 的源码工具包，然后将文件夹展开。展开后的目录结构上文已经有过详细的说明，直接把 Src 目录下的全部内容复制到 skype_dev_eg 工程的 Src 目录下即可。复制后的效果如图 11.8 所示。

从图 11.8 中可以看出，引入源代码后有些错误提示，在 Eclipse 平台的 Java 工程中，凡是以红色的 X 标记的文件都是有某种错误的文件。其实这个错误是缺少一些依赖的 Jar 包造成的，只需导入相关 Jar 包即可。这些依赖 Jar 包分别为 swt.jar 和 winp.jar，先将这两个包单独取出来，放在 D 盘目录下一个 javaLib 的文件夹里，路径为 D:/javaLib；导入此目

录中 Jar 包的方法如下。

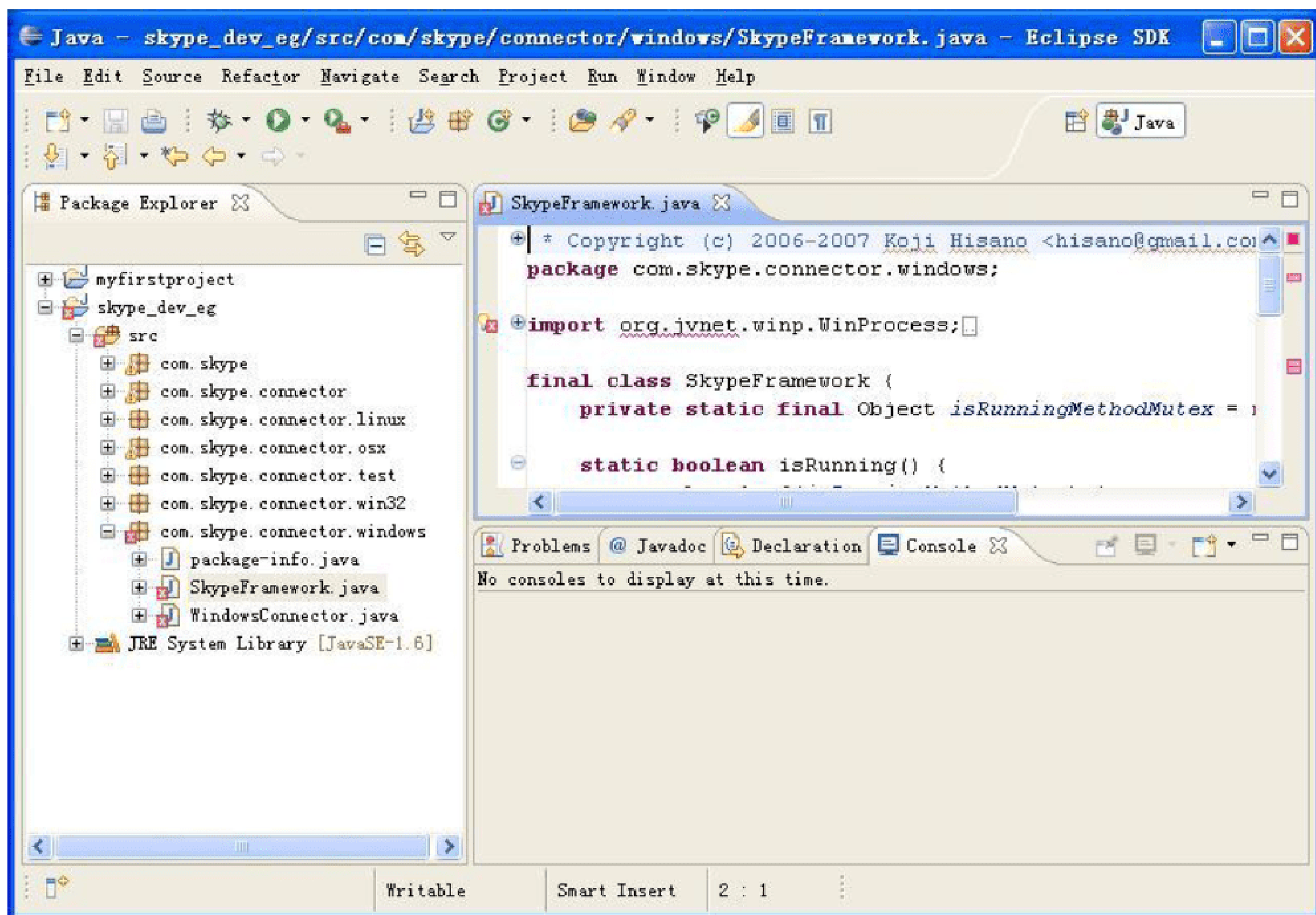


图 11.8　载入 Skype 源码后的工程示意图

注意： 在 Skype 的源代码包中，swt.jar 和 winp.jar 的位置在 lib/windows 目录下，导入源代码后需要将这两个包载入到构建路径下。在本书所带光盘中附有 Skype 开发包的源代码。

右击工程 skype_dev_eg，在弹出的快捷菜单中选择 Build Path | Configure Build Path 命令，操作过程如图 11.9 所示。

在打开的对话框中，选择 Libraries 标签，再单击左侧的 Add External JARs 按钮。在弹出的对话框中选择刚才新建的 javaLib 目录下的两个 Jar 文件，打开即可，操作如图 11.10 所示。

单击 OK 按钮，这样构建路径就加载完毕了。导入这两个包后，Eclipse 显示的错误信息就自动消除了，这样，Skype 源代码文件导入成功了。

3．新建一个 dev 包

为规范代码结构，示例中的所有代码应单独放到一个包里，与源代码区分开来，所以，在原有结构的基础上新建一个 dev 包，包名为 com.skype.dev。新建包的方法是选中 src 工程源文件目录右击，在弹出的快捷菜单中，选择 new|Package 命令，在弹出的对话框中输入相应包名称即可。新建后的效果如图 11.11 所示。

这样 Skype 开发的完整框架就搭好了，在程序测试的时候还需要与 Skype 客户端进行通信，所以在本机上测试还需安装 Skype 客户端软件。

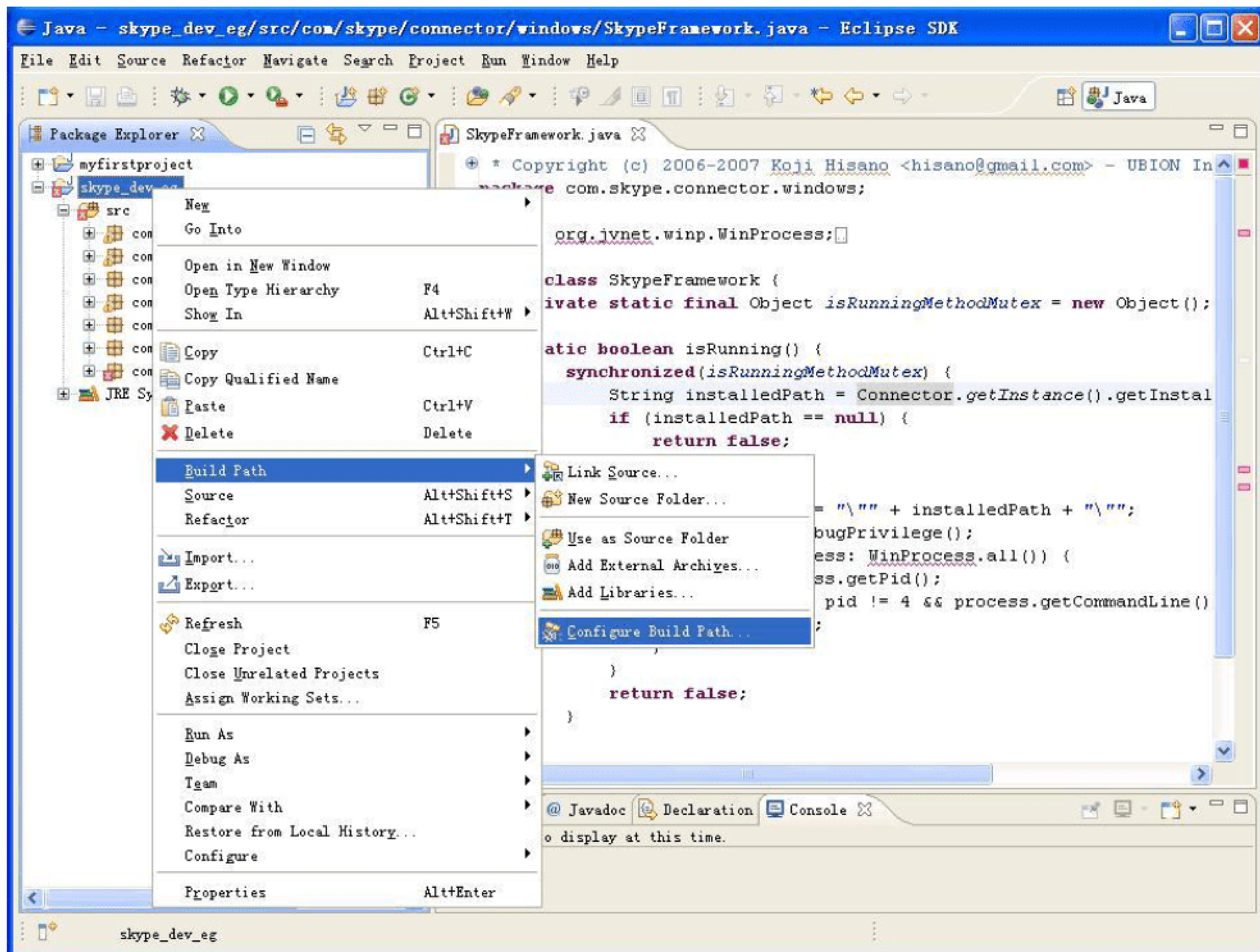


图 11.9　导入 Jar 包的操作示意图

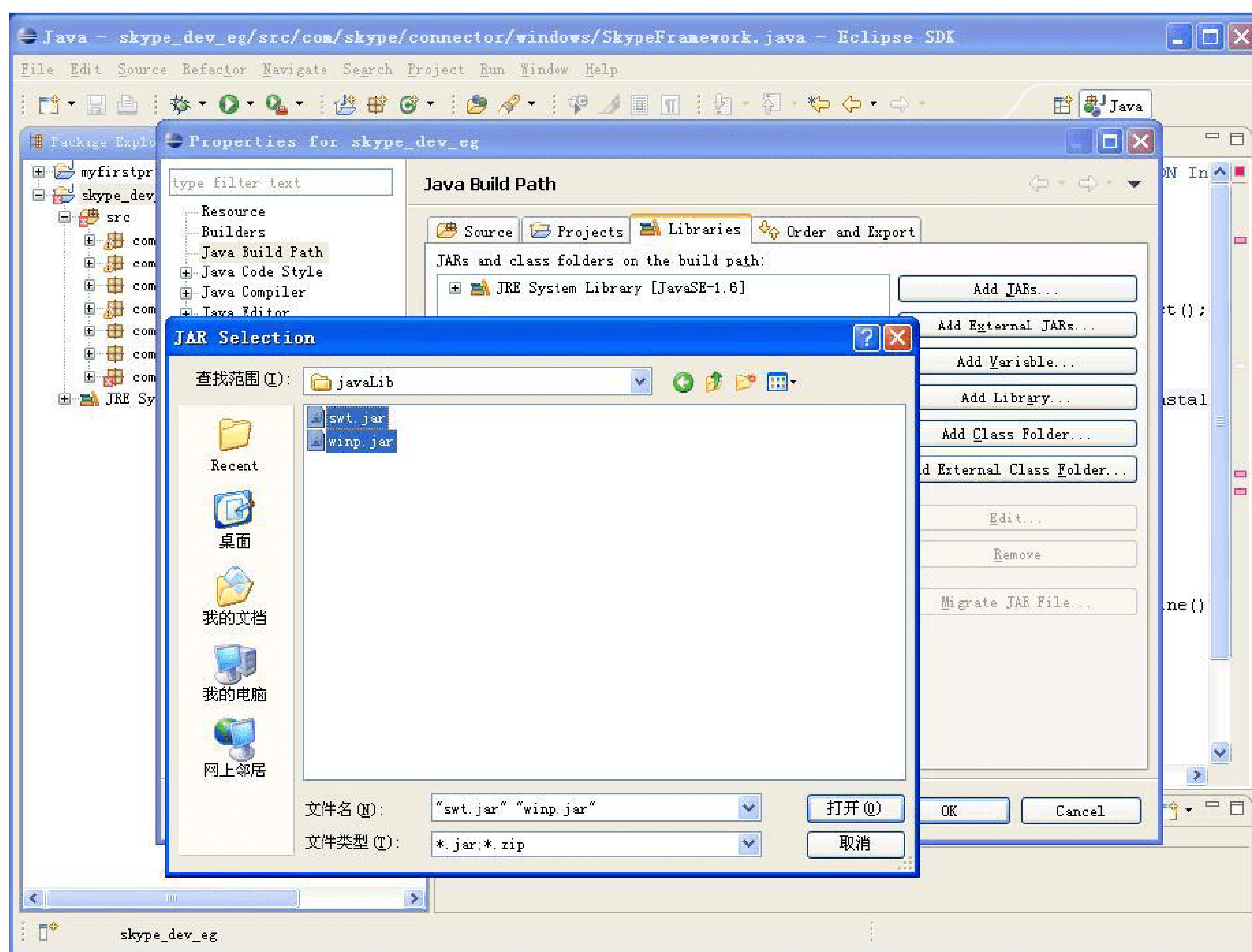


图 11.10　导入 Jar 包的过程

4．安装 Skype 客户端

Skype 客户端的安装、注册、登录及简单使用方法，在本书第 8 章有详细的讲解，请

读者自行参考相关内容，这里不再赘述。

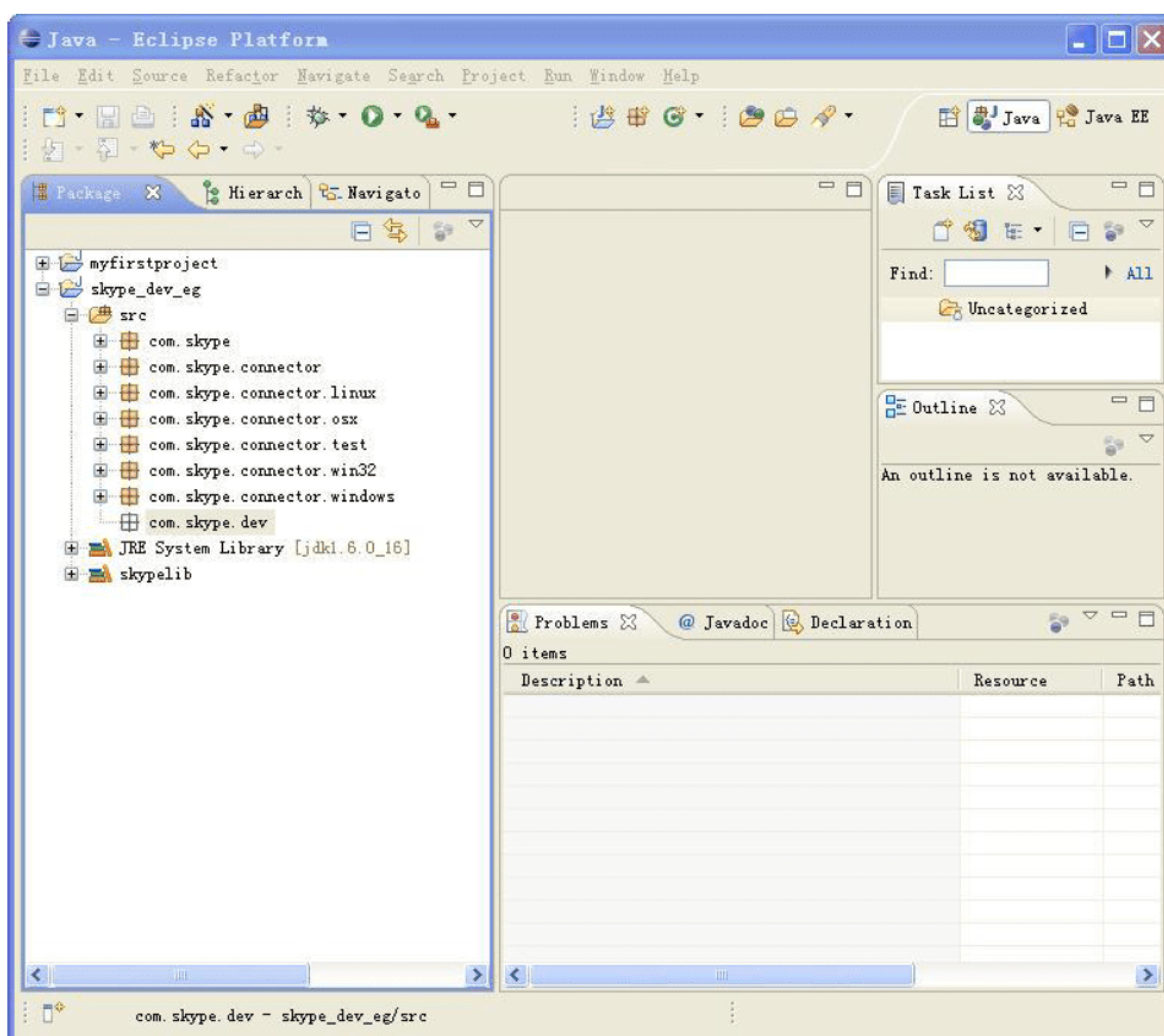


图 11.11　skype_dev_eg 工程框架结构图

需要注意的是，为了更好地测试 Skype 之间的通信、呼叫等，建议读者在两台主机上安装 Skype，注册两个账号并互相加为好友，这样，在测试的时候效果会更明显。

注意： Skype 不允许一台主机上同时运行两个 Skype 实例，所以最好将 Skype 安装在两台不同的主机上。

以上的准备工作完成以后，就可以进行 Skype 的应用开发了。在正式讲解 Skype 的一些基本应用开发之前，先讲一下 Skype 中应用程序之间的通信和命令发送的实现方法，因为 Skype API 两个核心的模块一个是通信，另一个就是命令协议，理解这两个原理对后文一些基本功能的实现很重要。

11.4.2　Skype 网络中应用程序之间的通信

应用程序之间的通信，简单地说就是你开发的应用程序（Application）可以利用 Skype 网络进行相互之间的通信。

在用 Skype4JAVA 开发 Application to Application 的应用时，只需要通过 Application 对象创建某一个特定名称的应用，然后连接到也创建了此应用的联系人，连接时会返回一个 Stream 对象，此对象可用于发送基于 TCP 和 UDP 的文本消息格式，然后通过注册的监

听器处理返回消息（此监听器即是应用程序相关的），这样，就实现了通过 Skype 网络体系的 AP2AP 应用。

注意：以上这段话中 AP2AP 的意思是，假如我是 UserA，我的 Skype 好友列表中有 UserB、UserC、UserD 这 3 个用户，我用 UserA 登录 Skype 以后，在 UserA 的客户端有一个基于 Skype4Java 开发的应用程序，为 Application A，这样我在 UserA 的一端运行 Application A 的时候，如果我的好友 UserB、UserC、UserD 他们之中任何一个或多个，在他们的客户端上也在运行着一个同样的 Application A 时，那么这两端的 Application A 之间就可以相互进行通信了，这就是 AP2AP 的意思。

下面通过例子简单介绍一下 AP2AP 的流程。这段程序将向所有的在使用同一个 Application 的联系人循环发送一条递增的 26 个字母的消息，并将对方利用 Stream 传递回来的文本消息打印到控制台。

注意：这种消息的格式是：比如第 1 条消息 a，则第 2 条消息为 bb，第 3 条为 ccc……依次类推，一直将 26 个字母发送完毕。

1．基本原理

首先，利用 Application 的 connectToAll()方法连接所有的联系人，这些联系人都是 Skype 好友列表中的用户，其中发送给 Skype 的文本消息如下：

```
ALTER APPLICATION <appname> CONNECT <friendid>
```

<appname>标识名，用来标识此 application 名称的，比如 ap2aptest<friendid>标识待连接好友的 Skype id。

connectToAll()方法将会返回一个 Stream 数组，对应每一个 friend id，都有一个唯一标识的 Stream 来维持通信。Stream 用来完成数据的传送，利用 Stream 的 write(String text)方法即可向联系人发送一个文本消息。

connectToAll()方法是 com.skype.Application 类的一个方法，方法原型是：

```
/**
     * 向所有的可进行连接的用户发出连接请求
     * @当连接失败或出错时，抛出 SkypeException 异常
     */
    public Stream[] connectToAll() throws SkypeException {
        return connect(getAllConnectableFriends()); //直接调用 connect()方法
                                                    返回结果
    }
```

在 connectToAll()方法中，只有一个核心的 connect(para)方法，这个方法中需要传入一个 getAllConnectableFriends()的对象作为参数。Connect()的核心实现代码请参考 Skype 开发包的源代码。

2．代码实现

AP2AP 通信的实现主要也就是调用了此 Connect()方法，它主要用来实现在 Skype 网

络中两个应用程序之间的通信，详细的实现代码请参考随书光盘所附源码。

【AP2AP.java 类示例，参考目录：\源代码\ch11\ch11_code\skype_dev_eg\】

以下是 AP2AP.java 类的主要方法说明：

```
package com.skype.dev;
import java.util.Arrays;
import com.skype.*;
public class AP2AP {
    public static void main(String[] args) throws Exception {
     //设置 Skype 的一些初始属性
     Skype.setDebug(true);
        Skype.setDeamon(false);
        //设定此应用的名字，并以此名字连接到服务器
        String name = AP2AP.class.getName();
        Stream[] streams = connectToServer(name);
        //定义传递消息的格式，26 个字母依次循环递增
        for (int i = 0; i < 26; i++) {
        //向每一个注册了相同应用的 SKYPE 实例发送文本消息，也是基于 SKYPE 开发应用的入
        口点之一
           //这里用了 Java1.5 的新特性，增强的 for 循环，对 stream 数组进行遍历
            for (Stream stream: streams) {
                stream.write(createData(i + 1, (char)('a' + i)));
            }
        }
        //程序暂停 5 秒，也就是 5 秒后中断连接
        Thread.sleep(5000);
        //再次对 stream 数组进行遍历，释放连接
        for (Stream stream: streams) {
            stream.disconnect();
        }
    }
```

在 AP2AP.java 类中，当需要实现两个应用程序通信的时候，还需要创建一个用于交互的消息，以下就是创建消息的代码。

```
   //创建消息的方法，需要传入一个整型的表示消息长度的参数，另一个 char 型的消息
   private static String createData(int length, char character) {
        //开辟一个新的字节数组空间，用于存储消息内容
        byte[] data = new byte[length];
        //调用 Arrays 类的 fill()方法，将指的 char 类型的值分配到字节数组中，需要类型
        转换
        Arrays.fill(data, (byte)character);
        //返回一个新的，包含消息的字符串
        return new String(data);
    }
```

以下是一个连接到服务器的方法，通过调用 Skype 开发包中 Skype 类的 addApplication()方法，传入一个字符串类型的名字参数以新建一个 Application 实例，在这个实例的监听器中创建连接，具体的实现方法如下：

```
/**
 * connectToServer 方法，用来连接到服务器，返回一个 Stream 数组
 */
private static Stream[] connectToServer(String name) throws SkypeException
{
```

```
        //通过 Skype 的 addApplication()方法，取得一个 Application 实例
        Application application = Skype.addApplication(name);
        //实现此 Application()实例的监听器
        application.addApplicationListener(new ApplicationAdapter() {
        //创建特名字的 Application，并注册入 Skype 实例中
        public void connected(Stream stream) throws SkypeException {
            //调用 printApplicationAndStreamName()方法，将连接后的 Stream 信息打
            印出来
                printApplicationAndStreamName("connected: ", stream);
            }
            //调用 disconnected()方法，释放连接
            public void disconnected(Stream stream) throws SkypeException {
                printApplicationAndStreamName("disconnected: ", stream);
            }
          private void printApplicationAndStreamName(String header, Stream
          stream) {
          stem.out.println(header + stream.getApplication().getName() + "-"
          + stream.getId());
            }
        });
        //向所有的好友列表中的用户发出连接请求，此方法返回 Stream 数组
        return application.connectToAll();
    }
}
```

3．结果测试

在两台主机上进行测试，Skype 客户端开启以后，运行 AP2AP。在程序第一次运行的时候，Java 程序会在内容与 Skype 客户端进行通信，这时 Skype 会有个提示，是否允许此通信过程，如图 11.12 所示。



图 11.12　应用程序与 Skype 通信时的提示信息

注意： *在本书案例的测试中，注册了两个 Skype 账号，分别为 skype4java 和 water_blue 286，这两个用户互相加为好友。*

在图 11.12 所示的情况下，直接允许即可。在应用程序运行后，如果另一个客户端没有对应的启动此应用程序，那么控制台会显示如下信息，如图 11.13 所示。

由图 11.13 中控制台所显示的信息可以看出，其中有一个命令和应答为：

```
-> SEARCH FRIENDS
<- USERS echo123, skye4java
```

这是向 Skype 发送查找好友的命令，返回当前 Skype 中的好友列表信息，与图 11.13 左侧 Skype 好友中的用户一致，整个程序此时处于一种暂停执行的监听状态。

当在另一个 Skype 客户端启动此 AP2AP 的应用程序执行时，双方就开始进行通信了。

交互的消息刚好就是在程序中设定的消息，如图 11.14 所示。

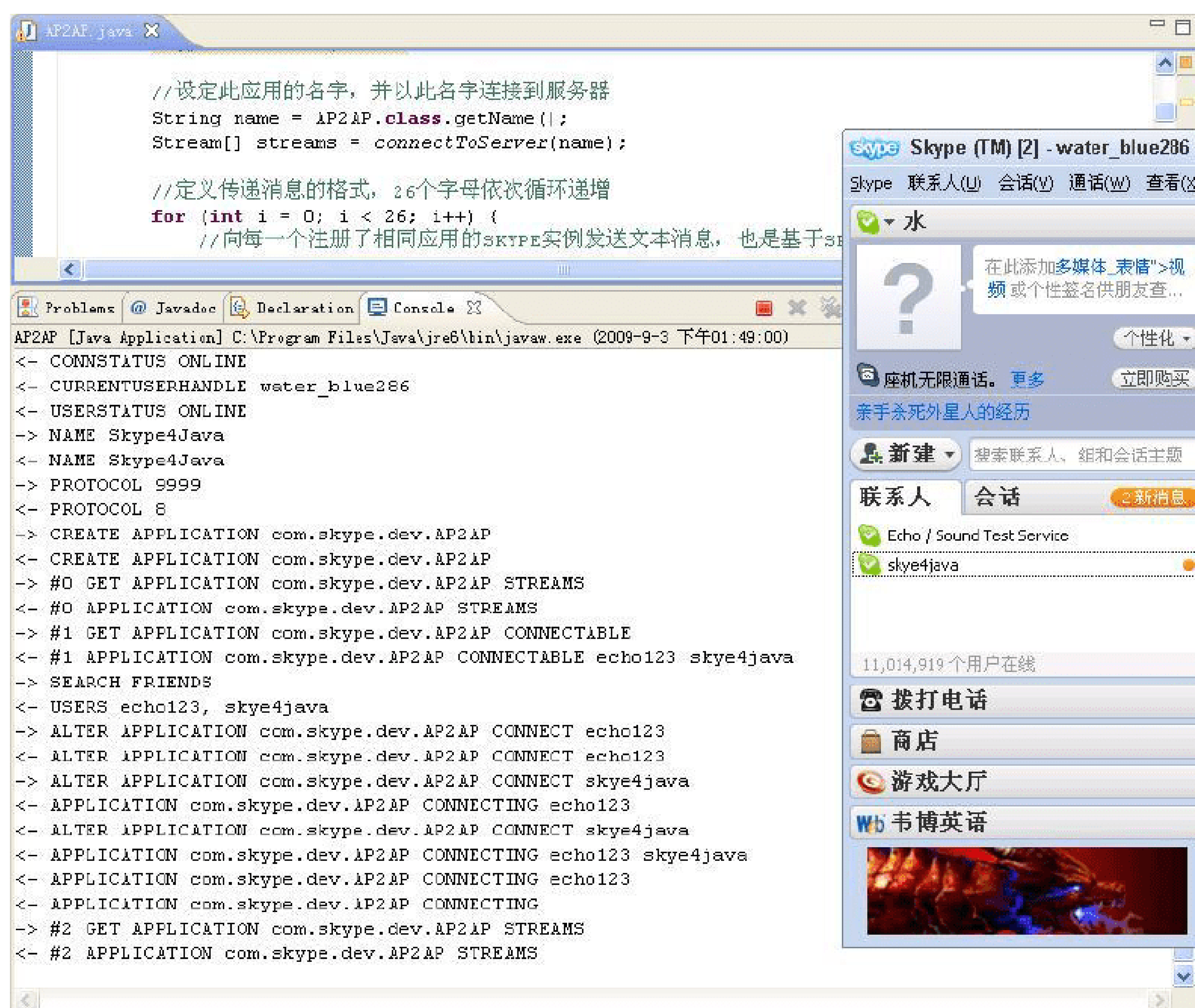


图 11.13　应用程序启动后控制台显示的信息

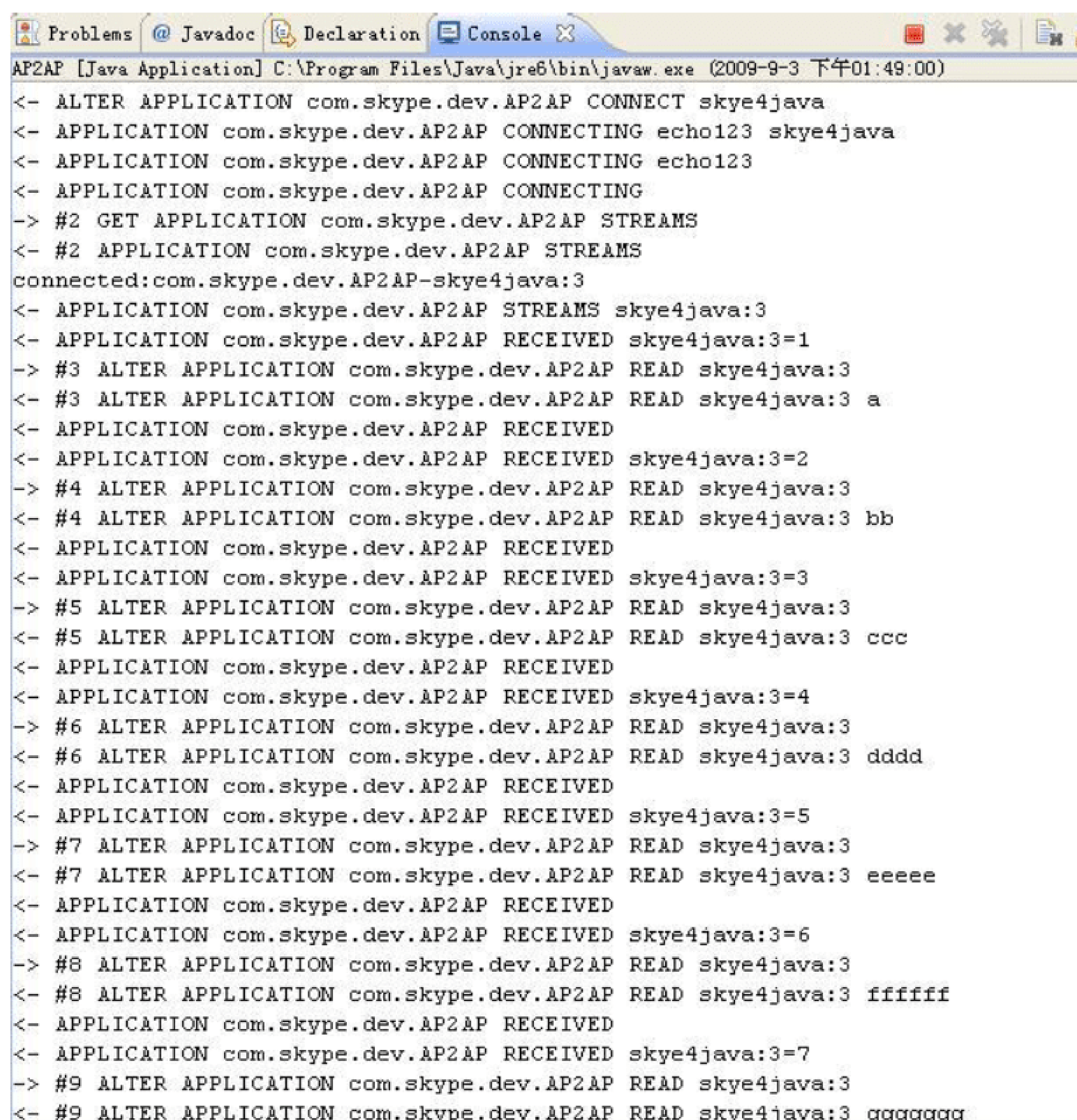


图 11.14　两个应用程序之间的通信消息展示

图 11.14 中，第一条消息为 connected：com.skype.dev.AP2AP-skye4java：3，证明当前的 Skype 客户端监听到好友 skye4java 的客户端启动的 AP2AP 应用程序，然后这两个应用程序就可以相互传递消息了。

在此应用程序中，还可以加一个 Stream 的监听器来对 Stream 的消息做出反应，如下：

```
stream.addStreamListener(new StreamAdapter() {
//此方法完成对于对方传送数据的处理，也是基于 SKYPE 开发应用的入口点之一
    public void textReceived(String receivedText) throws SkypeException {
        //要开发的业务逻辑
        System.out.println("what I can do……");
    }
    });
```

加上这个监听器后，一旦用户接收到来自于 Stream 的消息，即通过 textReceived（String receivedText）方法来处理。当需要开发自己的应用时，完全可以向 StreamAdapter 中添加方法，进行自己的业务逻辑开发，然后做相应的处理。因此在互为联系人的 Skype 用户两端运行这个程序，就完成了消息的发送和接收。

以上就是 Skype 客户端之间应用程序之间的通信。下面再讲解一下，如何通过应用程序向 Skype 发送命令。

11.4.3　Skype 网络中的基本命令发送

Skype 提供了几种形式的开发接口给不同的语言，也就形成了针对不同语言的开发包，但所有这些开发包都有一个共同点就是，它们都是通过相同的命令来操作 Skype 的。

对于 Skype4Java 来说，命令的发送都是通过执行 Skype4Java 的抽象连接层实现，这个连接层中有一个核心类 Connector.java，它在 Skype 源代码包中的 com.skype.connector 包下。在此类中，有一个核心的 execute()方法来执行各种命令。此方法的原型为：

```
rotected final String execute(final String command, final String[]
responseHeaders, final boolean checkAttached),
```

这个方法体里包含了保证与 Skype 处于连接状态，如果没连接则会自动连接上，激发各种事件及注册对该事件感兴趣的监听器。

注意： 向 Skype 发送的命令由命令标识符来标识，主要是在确认一个详细的请求命令以及响应消息时用的。每个命令及响应的消息 ID 是一样的，而且是唯一的。

关于 Skype 中的 Connector 类上文已有过说明，此类的完整源代码读者可以在 Skype 的开发包中查看，它有一个重要的 execute()方法，下面重点说明一下这个方法。

Connector 类中的 execute()方法，主要用于应用程序向 Skype 网络发送命令，这个方法执行命令发送的具体实现，方法内容如下：

```
protected final String execute(final String command, final String[]
responseHeaders, final boolean checkAttached) throws ConnectorException {
      //先打印一个提示信息，开始准备执行命令
      System.out.println("...准备执行命令...");
      //检验这次需要发送的命令，保证命令内容、响应消息头不为空
```

```
    ConnectorUtils.checkNotNull("command", command);
    ConnectorUtils.checkNotNull("responseHeaders", responseHeaders);
     //输出这次需要发送命令的有关内容
     System.out.println("command: ->"+command);
     //检查连接状态
      if (checkAttached){
       //在发送之前得先保证客户端与 Skype Client 处于连接状态
       assureAttached();
      }
      //共享锁，因为这是一种请求-响应式的交互，正在执行此命令时要锁住程序专门处理这
      个命令的发送
      final Object lock = new Object();
      //命令的响应
      final String[] response = new String[1];
      //连接监听器的定义
      ConnectorListener listener = new AbstractConnectorListener() {
     //当命令发送后(下面的)完全释放锁，客户程序将可以处理接收到的命令响应消息，查
     看此命令的执行结果
       System.out.println("response: <-"+message);
       //以下的方法用于处理接收到的消息，需要传入一个 ConnectorMessageEvent 对象
       public void messageReceived(ConnectorMessageEvent event) {
           //取得消息内容
           String message = event.getMessage();
           //将得到的消息打印输出
           System.out.println("response: <-"+message);
             //处理响应消息头，用增强的 for 循环，对响应头消息进行遍历
              for (String responseHeader : responseHeaders) {
                 //对响应头消息进行判断
                  if (message.startsWith(responseHeader)) {
                      response[0]=message;
synchronized (lock) {                          //对 lock 对象加锁
                          lock.notify();          //唤醒其他的线程
                      }
                      return;
                  }
              }
          }
      };
      //把上面定义的连接监听器注册到 Connector
      addConnectorListener(listener, false);
      //每次需要执行一个客户端命令时，都会调用 fireMessageSent 来激发所有注册进来
      的监听器
      fireMessageSent(command);
      //发送命令的过程将会获得共享锁
      synchronized (lock) {
          try {
              //调用不同子类的 sendCommand()方法来实现不同平台的消息发送
              sendCommand(command);
              //获取发送命令时的当前时间
              long start = System.currentTimeMillis();
              //获取发送命令超时的时间
              long commandResponseTime = getCommandTimeout();
              //对命令响应的时间加锁
              lock.wait(commandResponseTime);
              //对超时与否进行判断
              if (commandResponseTime <= System.currentTimeMillis() -
```

```
                start) {
                    //根据判断结果，对状态进行相应设置
                    setStatus(Status.NOT_RUNNING);
                    //抛出超时异常
                    throw new TimeOutException("The '" + command + "' command
                    failed by timeout.");
                }
            } catch (InterruptedException e) {
                //捕获到任何问题，则此次连接失败，抛出连接异常
                throw new ConnectorException("The '" + command + "' command
                was interrupted.");
            } finally {
                //不管连接成功与否，最终要移除监听器
                removeConnectorListener(listener);
            }
        }
        return response[0];                //返回响应信息
}
…
}
```

注意： Skype 开发包源代码中关于 execute()方法的实现，不是很好理解，为配合读者理解，笔者添加或者分解了一些原有的方法，所以和源代码有部分不一致的地方，但内部实现原理是完全一样的。如果读者理解了其内在的实现原理，完全可以自己重写一个 Connector 类。

下面就利用 Skype 的命令机制，通过应用程序向 Skype 发送一个简单的命令，并以此来验证一下 Skype 命令执行的流程。

在 com.skype.dev 包中，新建一个类，类名为 GetSkypeVersion，此类主要用来向 Skype 客户端发送命令，取得当前 Skype 版本的相关信息。本例源代码请参考：

【GetSkypeVersion.java 类示例，参考目录：\源代码\ch11\ch11_code\skype_dev_eg\】

```
package com.skype.dev;
import com.skype.Skype;
import com.skype.SkypeException;
/**
* @取出 Skype 客户端的版本信息
 */
public class GetSkypeVersion {
    public static void main(String[] args) {
        //捕获 SkypeException 异常
        try {
            //直接调用 Skype 的 getVersion()方法，此方法返回一个描述版本信息的字
                符串
            String skypeVersion = Skype.getVersion();
            //打印输出版本信息
            System.out.println("当前 Skype 的版本为：" + skypeVersion);
        } catch (SkypeException e) {
            //打印异常堆栈
            e.printStackTrace();
        }
    }
}
```

GetSkypeVersion 是一个 Eclipse 下的工程，直接在 Eclipse 中运行，就可以得到程序运

行的结果，结果显示如图 11.15 所示。

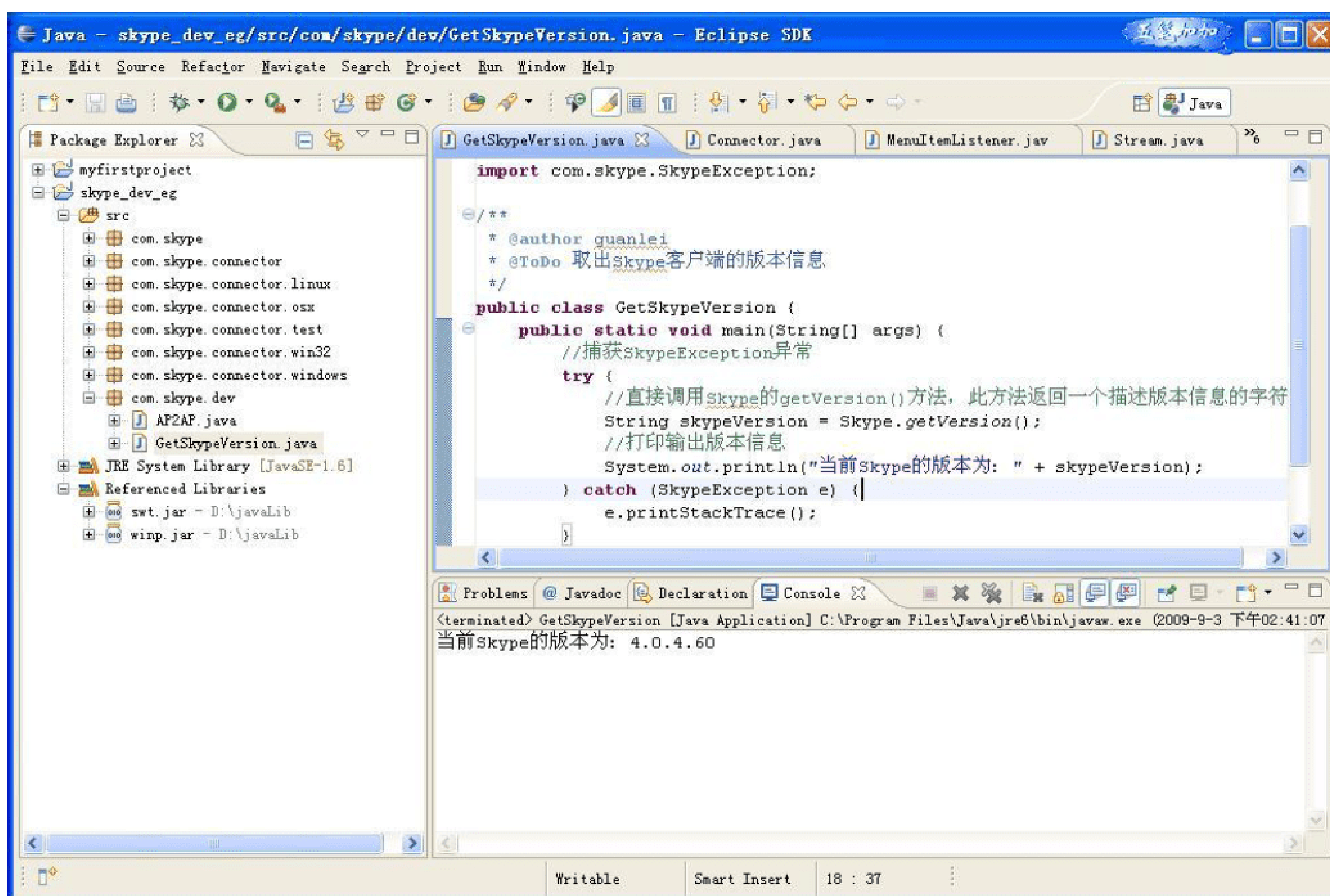


图 11.15　获取 Skype 版本信息的运行结果

图 11.15 是应用程序得到的 Skype 版本信息，Skype 本身的版本信息如图 11.16 所示。



图 11.16　Skype 版本信息显示

当前的版本为 4.0.4.60，通过图 11.15 与图 11.16 的对比，可以清楚地说明，通过应用程序发送命令，可以取得 Skype 的版本信息。

在 GetSkypeVersion.java 中，最关键语句是 Skype.getVersion()。在这个方法中，Skype 调用一个 Utils.getProperty("SKYPEVERSION");，在此方法中，getProperty 执行了下面的语句：

```
String response = Connector.getInstance().execute(command, response-
Header);
```

这样就通过 Connector 类的 execute()方法将 Skype 命令发送出去，同时此方法也返回相应的应答信息，这样就得到了 Skype 的版本信息了。

11.5　Skype 的基本应用开发

Skype 的基本应用程序都是以插件的方式附加到 Skype 客户端的。插件，简单地说就是一个由用户启动的普通应用程序。Skype 插件就是通过 Skype 的公用 API 开发的、从 Skype 的外部连接到 Skype 的客户端，并与客户端之间进行交互的应用程序。通过这些应用程序可以增加 Skype 的功能，可实现对 Skype 的各种操作。本节就重点讲一下 Skype 的基本应用开发。

注意： 针对 Skype 的特性而言，Skype 插件是一系列软件，它包括可以自动（或定时）发送信息和打电话，监视 Skype 发送和接收的消息，视频音频的录制，Skype 用户的搜索，等等，也就是说通过插件，几乎可以自动操作 Skype 的任意功能！

11.5.1　一个简单的 Skype 插件示例

在开发 Skype 插件之前，很多读者都会迷惑一个问题，应用程序是如何与 Skype 客户端联系起来的呢？

先看一幅截图，如图 11.17 所示，在 Skype 客户端界面上，选择“工具”|“更多功能”命令，发现“更多功能”这个选项是处于不可用状态。



图 11.17　Skype 的工具项菜单情况

再看下面这段代码，具体这段代码的含义后文会讲解，现在只要把它运行起来就行。在 skype_dev_eg 工程下的 com.skype.dev 包里，新建一个类，类名为 SkypeMenuTest.java，此类用来在 Skype 菜单中显示一个菜单项。

下面对 SkypeMenuTest.java 类中的主要方法进行说明，详细的、可运行的源程序源代码请参考随书所附光盘。

【源代码示例：\ch11\ch11_code\skype_dev_eg\src\com\skype\dev\SkypeMenuTest.java】

```
/**
 *本方法主要用来在 Skype 中附加一个按钮，然后通过这个按钮来控制应用程序。这样，应用程序
 *与 Skype 客户端之间就建立了一个联系的桥梁。主要目的是测试一下如何将一个应用程序附加到
 * Skype 客户端中。
 */
package com.skype.dev;
import com.skype.*;
//定义一个 SkypeMenuTest 类，用来测试通过应用程序来附加一个按钮到 Skype 客户端
public class SkypeMenuTest {
```

```
    public static void main(String[] args) throws Exception {
        //调用 Skype 类的 setDebug()和 setDeamon()方法,关于这两个方法的意义，请参考
        Skype 类
        Skype.setDebug(true);
        Skype.setDeamon(false);
        //定义一个 final 型的 MenuItem 对象，用来产生的一个按钮，并设定此按钮的名称
        final MenuItem item = SkypeClient.addMenuItem(MenuItem.
        Context.TOOLS, "Skype4Java menu", null, null, true, null, true);
        //在此按钮上加入监听动作，也就是单击这个按钮时的反应
        item.addMenuItemListener(new MenuItemListener() {
            public void menuItemClicked(MenuItemClickEvent event) throws
            SkypeException {
                //当此按扭被单击时，打印一条消息
                System.out.println("Skype4Java menu is clicked.");
                //动作完成后，释放
                event.getMenuItem().dispose();
            }
        });
    }
}
```

在 Eclipse 中运行上述代码，程序运行以后，再打开 Skype 客户端菜单“工具”|“更多功能”，这时，新的菜单出现了，如图 11.18 所示。

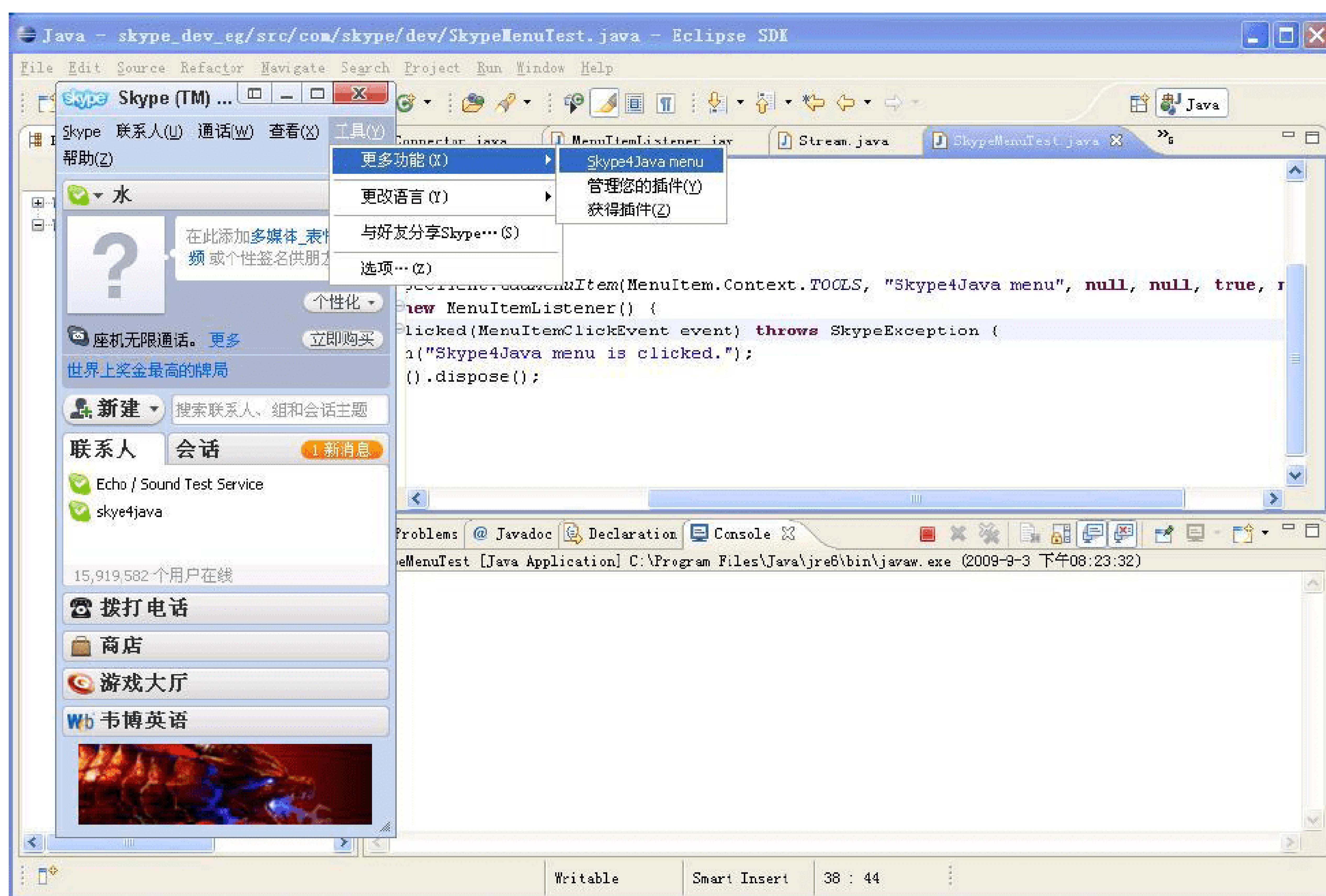


图 11.18　应用程序运行后添加的新的菜单项

Skype 客户端中出现的新的菜单名称就是程序中的 Skype4Java menu。当单击这个按钮的时候，就会执行程序中的 menuItemClicked()方法，在控制台打印出 Skype4Java menu is clicked 的消息，读者操作一下便知。

通过以上的演示，可以说明两个问题：

- 第一，外部的应用程序可以实现对 Skype 客户端本身的控制，比如添加一个菜单项。

- 第二，在控制 Skype 客户端的时候，可以加入自己的业务逻辑，比如单击菜单时打印一条消息。

这是最简单的对 Skype 插件工作方式的理解，可以通过程序写多个菜单项，这些菜单项都可以显示在 Skype 客户端上。然后再在菜单项中加入各种业务方法，那么用应用程序来控制或操作 Skype 就是一件很容易的事了。

再说这段代码，这段代码的核心功能就是在 Skype 客户端中添加一个按钮，然后在这个按钮上加一个监听器，当单击这个按钮的时候，打印一条消息。上述实现很简单，主要就是调用了 SkypeClient 类的 addMenuItem()方法。

在讲解 Skype API 命令的时候，有一个示例，就是如何在 Skype 客户端添加一个菜单项，命令内容如下：

```
-> CREATE MENU ITEM test01 CONTEXT contact CAPTION "TEST 01" ENABLED true
<- MENU_ITEM test01 CREATED
```

通过这个命令就可以在 Skype 客户端创建一个名为 TEST 01 的菜单项。

而 addMenuItem()方法，它的实现原理就是向 Skype 发送了这条命令，addMenuItem()方法说明如下。

addMenuItem 方法，在类 SkypeClient.java 中定义，详细的源代码请参考随书所附光盘。

【源代码示例：\ch11\ch11_code\skype_dev_eg\src\com\skype\SkypeClient.java】

```
//通过命令在 Skype 客户端中新建一个按钮，这是 Skype4Java 开发包的内部实现机制，已经将命令都封装好了，在实际的应用中，只需调用这个方法就可以新建一个按钮
public static MenuItem addMenuItem(MenuItem.Context context, String
caption, String hint, File iconFile, boolean enabled, String targetSkypeId,
boolean multipleContactsEnabled) throws SkypeException {
        try {
            MenuItem menuItem = MenuItem.addMenuItem(context, caption, hint,
            iconFile, enabled, targetSkypeId, multipleContactsEnabled);
            //在命令层中，通过 Skype 命令实现新建一个按扭的实现方法
            String command = "CREATE MENU_ITEM " + menuItem.getId() + " CONTEXT
            " + context.name().toLowerCase() + " CAPTION \"" + caption + "\"";
            //以下是对每一个命令格式、内容的判断，是一种编程的约定，是 Skype 底层的实现方法，只需要用程序实现此命令的格式和内容，然后发送出去即可，读者可以不需要深入理解
            if (hint != null) {
                command += " HINT \"" + hint + "\""; //对每一个命令进行判断
            }
            if (iconFile != null) {
                command += " ICON \"" + iconFile.getAbsolutePath() + "\"";
            }
            if (!enabled) {
                command += " ENABLED false";
            }
            if (targetSkypeId != null) {
                command += " CONTACT TYPE FILTER \"" + targetSkypeId + "\"";
            }
            if (!multipleContactsEnabled) {
                command += " ENABLE MULTIPLE CONTACTS false";
            }
    //对命令消息的反馈结果，根据这个结果来反馈给用户，按钮是否成功新建
    String responseHeader = "MENU_ITEM " + menuItem.getId() + " CREATED";
```

```
    String response = Connector.getInstance().execute(command,
    responseHeader);
            Utils.checkError(response);
            return menuItem;
        } catch (ConnectorException e) {
            Utils.convertToSkypeException(e);
            return null;
        }
    }
```

仔细分析这个方法就会发现，整个方法体就是在判断并执行“创建 Skype 菜单项”的命令过程。这个菜单创建完成以后，只要再为这个菜单项添加一个监听器，那么所有的针对菜单项的操作都会被监听到。然后再在这个监听器中加入操作的业务方法，在你操作这个菜单项时，你的业务方法就会被执行了。

11.5.2　Skype 命令测试工具开发

在具体讲解 Skype 的业务方法之前，先讲一下如何用程序实现一个测试 Skype 命令的小工具。在 Skype API 命令一节里也说过，Skype 开发网站上提供有一个免费的 Skype-Tracer 小程序。下面就讲一下如何利用 Skype4Java 的工具包，编程实现一个类似的 SkypeTracer 程序。

在 Eclipse 下，skype_dev_eg 工程的 com.skype.dev 包里，新建一个类名为 SkypeTracer.java 的程序，用于实现一个 Skype API 命令测试的小工具。

注意：此程序是一个可视的界面程序，借助 org.eclipse.swt 包来实现，所以需要先引入此包的相关类。

以下只对 SkypeTracer.java 中的主要方法进行说明，详细的源代码请参考随书所附光盘。

【源代码示例：\ch11\ch11_code\skype_dev_eg\src\com\skype\dev\SkypeTracer.java】

```
/**
 *SkypeTracer 类是一个用来测试 Skype 命令的小程序，有一个简单的命令行输入界面，程序运
 *行后根据初始条件建立到 Skype 客户端的连接，用户可以在此界面中输入与 Skype 通信的命令，
 *并显示来自 Skype 的应答信息，类似于一个命令行控制终端，如 shell
*/
package com.skype.dev;
import java.io.*;
import org.eclipse.swt.*;
import com.skype.*.*;
import com.skype.connector.osx.OSXConnector;
//开发一个测试 skype 命令的小工具，类似一个 Shell，是用户与 Skype 通过命令进行交互的
中介
public class SkypeTracer extends Shell {
    public static void main(final String args[]) throws Exception {
            //直接调用 OSXConnector 类的 disableSkypeEventLoop()方法，请参考
            OSXConnector 类
        OSXConnector.disableSkypeEventLoop ();
```

```
        //定义一个 final 型的 Display 对象，用来显示相关信息
        final Display display = Display.getDefault();
        // 通过调用 SkypeTracer 的构造方法，new 一个 SkypeTracer 的实例，shell
        SkypeTracer shell = new SkypeTracer(display, SWT.SHELL_TRIM);
        shell.layout();                         //定义 Shell 的布局
        shell.open();                           //打开 Shell
        while (!shell.isDisposed()) {
            if (!display.readAndDispatch()) {
                display.sleep ();
            }
        }
    }
    //构造方法，也是核心的用来实现 SkypeTracer 功能的方法，需要传一个 Display 对象，
    一个 int 类型的，用来表示显示风格的整数
    public SkypeTracer(Display display, int style) throws
    ConnectorException {
        super(display, style);                  //调用父类的构造方法
        createContents();          //调用核心的 createContents()方法，下文会讲到
    }
```

SkypeTracer 是用多线程的方式来实现的，因为可以同时开启多个 SkypeTracer 进程，用来跟 Skype 客户端通信，以下是多线程的具体实现过程。

```
//供 SkypeTracer 构造方法调用的 createContents()方法实现过程
private void createContents() throws ConnectorException {
    setText("Skype Tracer");                    //设置标题
    setSize(400, 300);                          //设置大小
    final GridLayout gridLayout = new GridLayout();
                                                //定义一个 final 类布局变量
    gridLayout.numColumns = 2;                  //整个布局分为两列
    setLayout(gridLayout);                      //设置布局方式
    final Text fromSkype = new Text(this, SWT.V_SCROLL | SWT.MULTI |
    SWT.READ_ONLY | SWT.BORDER);
    fromSkype.setLayoutData(new GridData(GridData.FILL, GridData.FILL,
    true, true, 2, 1));
    //以下是多线程的实现过程，这里用到的 Java 匿名内部类的实现方式，有多个嵌套的内
    部类
    new Thread() {
    //重写 run()方法
        public void run() {
           //多线程的方法体，取得一个 Connector 的实例
            Connector.getInstance ().setDebugOut(new PrintWriter(new
            Writer() {
              //通过一个内容类，定义一个 write()方法，用于输出命令
              public void write(char[] cbuf, int off, int len) throws
              IOException {
                    //定义一个 final 型的 String 变量，用于表示写的内容
                    final String appended = new String(cbuf, off, len);
                    //方法内容通过实现 Runnable 接口，实现多线程
                    Display.getDefault ().asyncExec (new Runnable () {
                        //实现 run()方法
                        public void run() {
                            if (!fromSkype.isDisposed()) {
                                fromSkype.append (appended);
                            }
```

```
                    }
                });
            }
            try {
                Connector.getInstance ().setDebug(true);
            } catch(ConnectorException e) {
            }
        }
    //调用 start()方法，启动多线程程序
    }.start();
}
```

注意：以上只是对 SkypeTracker 类中的主要方法进行说明，读者在学习这个程序的时候，一定要结合源代码并实际运行来理解。在这个程序中，结构比较复杂，尤其注意理解多线程的实现方式、嵌套的内部类等知识点。

在这个程序中，SkypeTracer 继承了 SWT 包里的 Shell 类，用到了 OSXConnector 类里的一些方法，内部实现都是用多线程的方式进行命令的发送和接收的。想了解这些方法的详细信息请参考源代码，这里就不再详解。

完成代码后，运行程序，使用效果如图 11.19 所示。

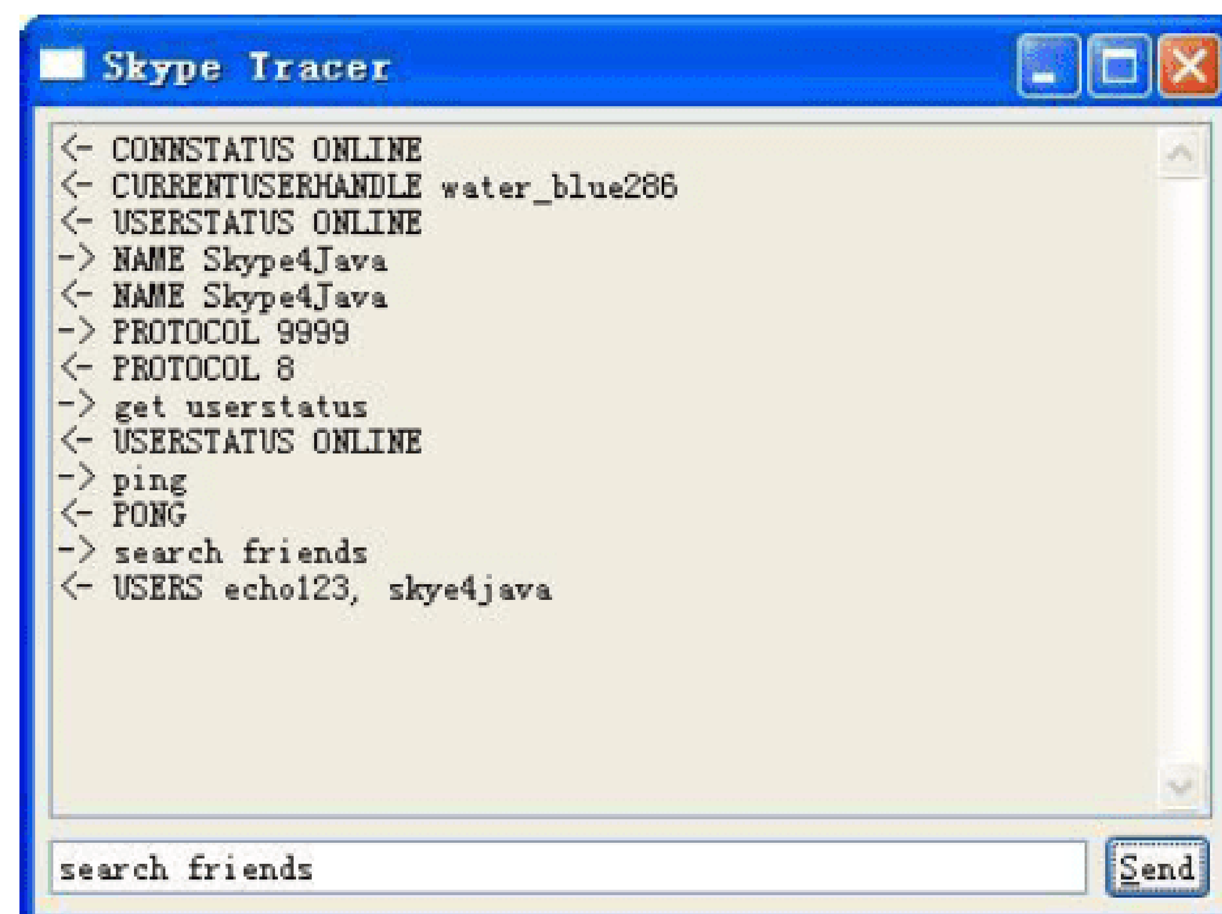


图 11.19　Skype Tracer 程序的界面截图

通过这个小程序，在开发 Skype 应用的时候，可以用它来简单地测试一个命令的内容、格式等是否正确，在程序开发的时候比较实用。

11.6　Skype 基本业务功能的实现

11.5 节讲了如何把一个应用程序附加到 Skype 中，也介绍了如何用一个小工具来测试 Skype API 的命令，然而真正的 Skype 插件的应用的还在于能控制 Skype 的各种操作，也就是 Skype 的各种业务方法，如呼叫、会话、自动应答等。本节将开始重点讲 Skype 应用中一些业务方法的开发。

11.6.1　产生一个呼叫

在客户端产生一个呼叫，意思是，通过应用程序的业务方法来操作 Skype 客户端，让 Skype 自动呼叫某个在线好友。

Skype4Java 的工具包中，在 Skype 类中提供了一个 Call()方法，主要用于产生一个呼叫。在这个 Call()方法中，需要传入一个 String 类型的 SkypeID，就是要呼叫的对象，然后返回一个 Call 对象。在 Call 对象中详细定义了各种呼叫的属性、状态及操作方法等。

在使用 Call()方法时，假如要呼叫好友列表中一个叫 UserA 的好友，那么只需调用 Call(UserA)命令即可。也就是说，在 Call()的方法中，传入 UserA 的 ID 号就能创建一个对

UserA 用户的呼叫。

1．代码实现

Call()方法的核心代码如下，此代码参见 Skype 源代码包中 com.skype 包下的 Skype.java 类里的 Call()方法。

【源码示例：\ch11\ch11_code\skype_dev_eg\src\com\skype\Skype.java】

```
//Skype4Java 开发包中 Skype 类中的 Call()方法说明
public static Call call(String skypeId) throws SkypeException {
  //调用 Utils 类的 checkNotNull()方法，对 skypeid 进行检查
     Utils.checkNotNull("skypeIds", skypeId);
     try {
          //定义响应头的字符串
          String responseHeader = "CALL ";
          //通过 Connector 类的 executeWithId()方法执行呼叫的命令，命令中有一个
          SkypeID 参数，就是要呼叫的用户 ID
         String response = Connector.getInstance().executeWithId("CALL "
         + skypeId, responseHeader);
         //调用 Utils 类的 checkError ()方法，对响应结果进行检查
         Utils.checkError (response);
         //对 response 字符串进行处理，取得 ID 信息
         String id = response.substring(responseHeader.length(),
         response.indexOf(" STATUS "));
         //调用 Call 类的 getInstance()方法，传入 ID 参数，返回一个 Call 对象实例
         return Call.getInstance(id);
        //捕获并处理相关异常
     } catch (ConnectorException e) {
         Utils.convertToSkypeException (e);
         return null;
     }
 }
```

在这段代码中，最核心的逻辑，就是调用 Connector 的 executeWithId()方法，来执行一个呼叫命令。这个方法在上文已经详细地讲过，在这个方法中需要传入两个参数，一个命令参数（String Command），另一个是响应头参数（responseHeader）。Connector 类以及 executeWithId 方法都已说明，不明白的地方请参阅上文的相关内容。

有了以上的说明，就可以很清楚地知道在 Skype 网络中通过应用程序创建一个呼叫的整个流程，真正的实现代码也很简单。

在 skype_dev_test 工程的 com.skype.dev 包中新建一个类，类名为 MakeCall.java，用于在 Skype 客户端创建一个呼叫。以下是 MakeCall 类中主要方法的说明，关于此类的源代码请参考随书所附光盘。

【源代码示例：\ch11\ch11_code\skype_dev_eg\src\com\skype\dev\MakeCall.java】

```
/**
* MakeCall 类，用于产生一个 Skype 呼叫。MakeCall 类在运行的过程中，会根据指定的呼叫
*参数向 Skype 客户端好友列表中的一个用户发出一个呼叫，具体实现如下：
*/
package com.skype.dev;
import com.skype.Skype;
```

```
//定义一个 MakeCall 类，用于在 Skype 客户端产生一个呼叫
public class MakeCall {
public static void main(String[] args) throws Exception {
        //需要用户输入参数才能运行，这里是对用户输入的参数进行判断
        if (args.length != 1) {
        //如果没有参数输入，或是用户输入有误，则打印提示信息，程序返回
        System.out.println("说明：请输入程序启动运行参数：参数为 'skype_id'，要
        呼叫的用户 ID");
            return;
        }
        //程序运行时，用户只需输入要呼叫的用户的 ID，程序即可自动产生一个针对此 ID 的
        呼叫
        System.out.println("呼叫的用户为：" + args[0] + "，呼叫时间为：" + Utils.
        showCurrentTime ());
        //调用 Skype 类的 call()方法，接收参数，产生呼叫
        Skype.call(args[0]);
    }
}
```

注意： 在此段程序中调用了一个 showCurrentTime()方法。为了用时间对比进行测试，程序中用此方法来显示当前时间，这个方法在 com.skype.dev.Utils 工具类中，是一个非常简单的实现，读者一看就能明白。

2．程序的验证和运行

运行 MakeCall 类，需要在主程序中传入相关参数，这个参数就是 Skype 客户端好友列表中的用户 ID。运行程序的过程如下。

打开 Skype 客户端，找到一个在线好友的 ID，如图 11.20 所示，就是当前这个 Skype 客户端中的用户情况。以图 11.20 在线用户 skye4java 为例来说明，如何通过 MakeCall()对 Skype 客户端中的 skye4java 用户进行呼叫。保持 Skype 客户端处于运行状态，在 Eclipse 中选择类文件 MakeCall.java 右击，在弹出的快捷菜单中选择 Run as|Run Configurations 命令，弹出如图 11.21 所示的对话框。

在图 11.21 所示的这个界面中，是 Main 标签所显示的情况。保证 Project 文本框中的名称为 skype_dev_test，Main class 文本框中的名称，为 com.skype.dev.MakeCall，也就是 MakeCall 类所在路径的全称。

在图 11.21 所示的对话框中，选择“（X）＝Arguments”标签，会显示如图 11.22 所示的界面，即输入参数对话框。在 Program arguments 区域中，输入要传入的参数，本例运行的类需要传入一个 SkypeID 号（即在线好友的 ID 号）。已经确定了当前 Skype 客户端中的 skye4java 用户在线，所以在参数列表中输入 skye4java，输入后的效果如图 11.22 所示。

输入完毕以后，单击图 11.22 中的 Run 按钮，开始运行程序。程序运行后，Skype 客户端就会向 skye4java 用户发出一个呼叫，如图 11.23 所示。

而在 skye4java 用户一端就会接收到这个呼叫，如图 11.24 所示。

通过图 11.23 与图 11.24 的对比，可以清楚地看到，在图 11.23 中，应用程序在 22:18 时，从 Skype 客户端 water_blue286 向好友 skye4java 发送一个呼叫。在图 11.24 中就可以看到在客户端 skye4java 的历史消息中，清楚地显示着，在 22:18 时的时候，接收到了来

自 water_blue286 的呼叫请求。



图 11.20　Skype 客户端中在线用户情况

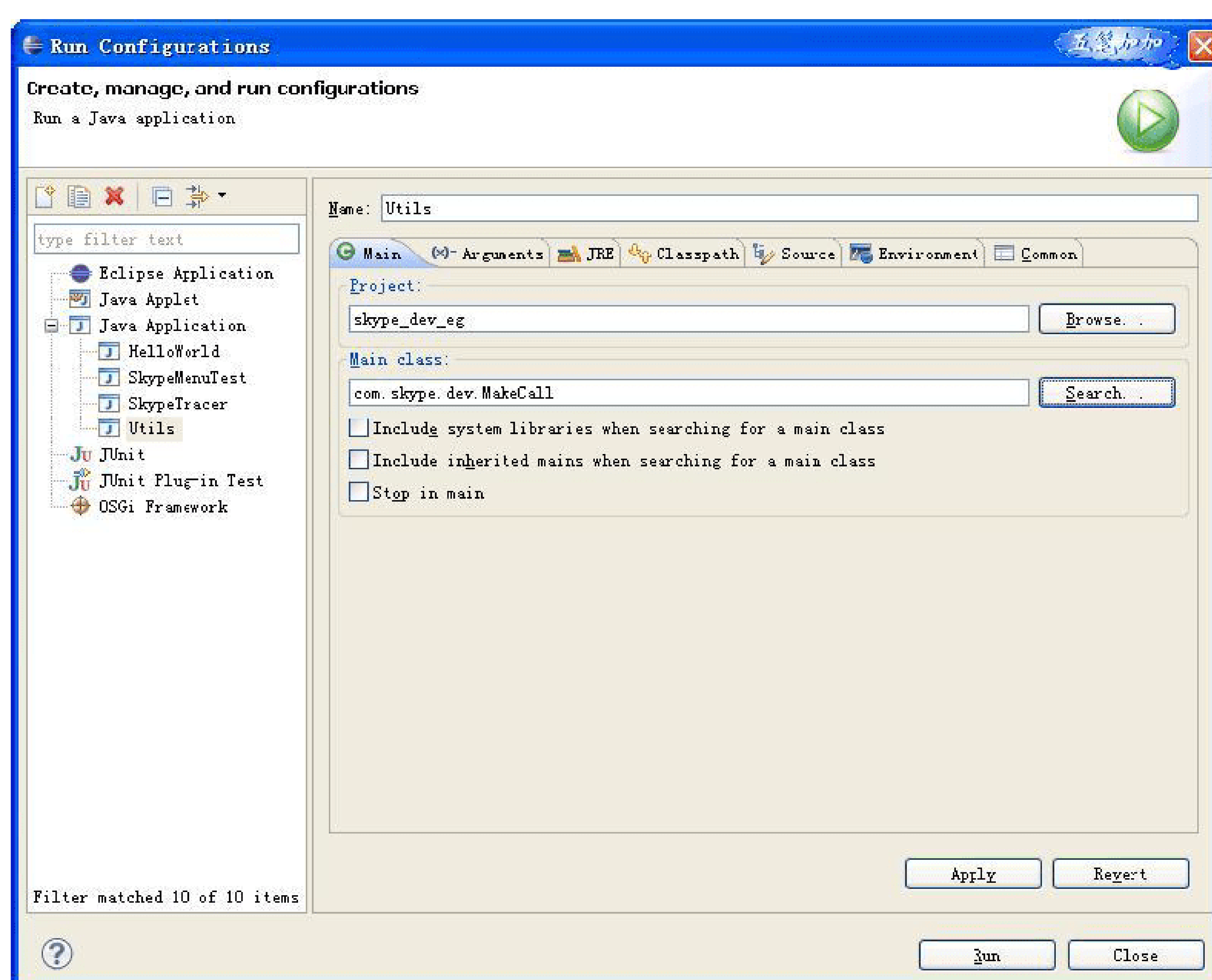


图 11.21　Java 程序运行是参数输入对话框

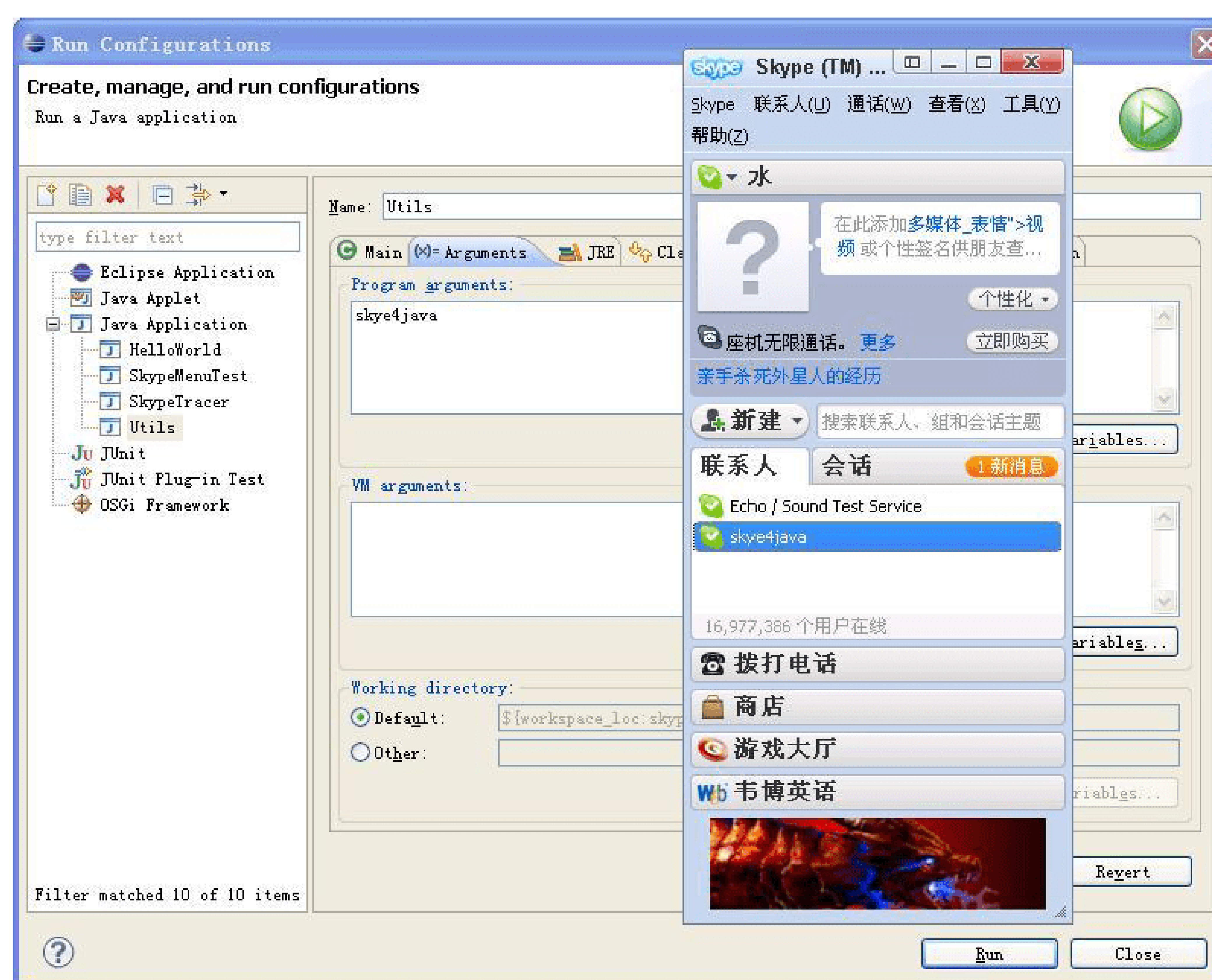


图 11.22　程序运行的参数输入示意图

由此可以证明，通过 Skype4Java 工具包开发的 Java 应用程序，通过调用 Call 方法，可以实现对 Skype 客户端的呼叫操作。这样，就完成了创建一个呼叫的业务方法。

以上的例子只是 Skype 应用开发中一个非常简单的应用，通过这些例子要重点掌握应用程序与 Skype 客户端之间的通信流程和使用方法。

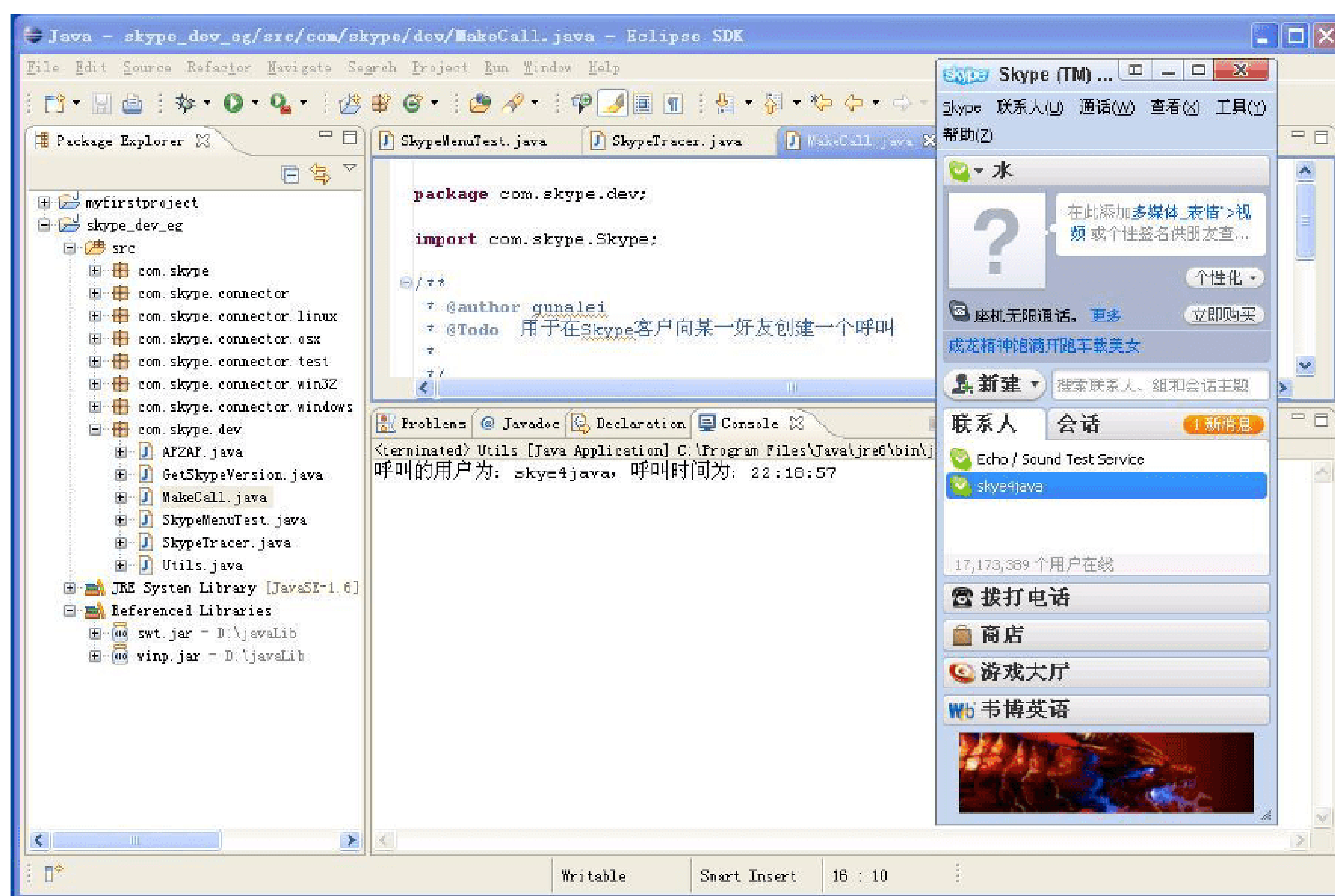


图 11.23　应用程序呼叫的结果示意图

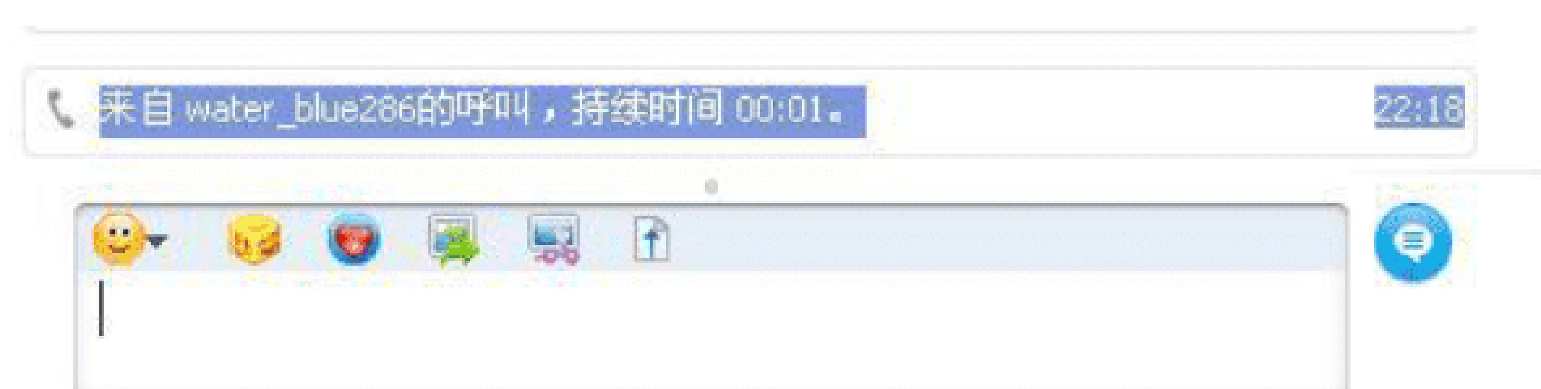


图 11.24　在 skype4java 用户端接收到的呼叫请求

11.6.2　发送聊天消息

向 Skype 客户端发送会话消息，意思是，通过应用程序，向 Skype 的某一好友发送聊天消息，而此好友所在的 Skype 客户端则能接收到这个消息。

1．实现机制

在 Skype 类中，提供了一个 Chat()方法来实现应用程序向 Skype 客户端发送聊天消息。Chat()与 Call()方法一样，接收一个 String 类型的 SkypeID 参数，返回一个 Chat 对象，Chat 对象定义了所有关于 Chat 的属性、状态和操作方法。下面就 Chat()方法中主要的代码进行说明，详细的源代码请参考 Skype 类中的 Chat()方法部分。源代码请参考随书所附光盘。

【源代码示例：\ch11\ch11_code\skype_dev_eg\src\com\skype\Skype.java】

注意：关于 Chat()方法的源代码，在 Skype 源码包的 com.skype.Skype.java 类下，这里只说明其部分代码。

```
/**
 *Chat()方法主要用来实现一个会话，在执行过程中与 Call()方法类似，也需要传入一个
  SkypeID
 *作为参数，Chat()方法会根据这个 ID，向拥有此 ID 的 Skype 用户发起一个会话。
 */
```

```
   public static Chat chat(String skypeId) throws SkypeException {
       try {
           //定义 Chat 命令的响应头信息
           String responseHeader = "CHAT ";
           //创建一个聊天会话的命令，返回一个字符串的应答信息
           String response = Connector.getInstance().executeWithId("CHAT
           CREATE " + skypeId, responseHeader);
           //调用 Utils 的 checkError()方法，对响应信息进行检查
           Utils.checkError(response);
           //对 response 的应答字符串进行处理，取出 ID 信息
           String id = response.substring(responseHeader.length(),
           response.indexOf(" STATUS "));
           //调用 Chat 类的 getInstance()方法，传入 ID 参数，返回一个 Chat 对象
           return Chat.getInstance(id);
           //捕获并处理异常信息
       } catch (ConnectorException e) {
           Utils.convertToSkypeException(e);
           return null;
       }
   }
```

执行 Skype.Chat()方法以后，就会得到一个 Chat 对象。要发送一条聊天消息，还需要调用 Chat 对象中的 send()方法，send()方法完整代码如下。

注意： send()方法的源代码，在 Skype 源码包中的 com.skype.Chat.java 类中，读者可找到此类，自行查看。

```
/**
 *一个完整的会话聊天过程，除了要建立一个必须的 Chat 对象之外，还需要完成消息的发送，
 *因为聊天是通过文本来传递信息的，只有将这个文本信息发送出来，才能实现并完成会话聊天
 *所需的基本功能，这也是不同于 Call 的地方，需要对比理解。
 */
 //以下就是 send()方法的具体实现，也要接收一个参数，这个参数就是要发送的消息内容
  public ChatMessage send(String message) throws SkypeException {
      try {
          //定义一个发送消息的响应头信息，也就是 Skype API 中发送消息所需的命令格式
          String responseHeader = "CHATMESSAGE ";
          //通过 Connector 实例的 executeWithId()方法执行发送消息命令，得到响应
            信息
          String response = Connector.getInstance().executeWithId
          ("CHATMESSAGE " + getId() + " " + message, responseHeader);
          //对应答信息进行错误检查
          Utils.checkError(response);
          //对响应信息字符串进行处理，取得消息 ID
          String msgId = response.substring(responseHeader.length(),
          response.indexOf(" STATUS "));
          //调用 ChatMessage 的 getInstance()方法，取得消息实例，传入消息 ID 返回
            会话对象
          return ChatMessage.getInstance(msgId);
          //捕获并处理异常
      } catch (ConnectorException e) {
          Utils.convertToSkypeException(e);
          return null;
      }
  }
```

在上述所讲的方法中，整个执行流程很简单，Send()方法体接收一个 String 类型的消息字符串作为参数。通过 Connector 实例的 executeWithId()方法，执行发送消息的命令，最后返回一个 ChatMessage 对象，这样，就可以把消息从 Skype 网络中发送出去了。

2. 代码实现

明白了以上消息发送机制，就可以编程实现一个应用程序来向 Skype 好友发送聊天消息了。在 skype_dev_eg 工程的 com.skype.dev 包中新建一个类，类名为 SendChatMessage，用于在 Skype 客户端发送一个聊天消息。以下是发送聊天消息的方法说明，此类的源代码请参考随书所附光盘。

【源代码示例：\ch11\ch11_code\skype_dev_eg\src\com\skype\dev\SendChatMessage.java】

```
 /**
  * SendChatMessage 类，主要用来实现应用程序通过 Skype 网络，向 Skype 客户端的好友列
  * 表中某一指定用户发送聊天消息的功能。当执行应用程序后，根据用户指定的参数，就可以把
  * 一条消息发送给另一个 Skype 客户端用户，具体实现如下：
  */
package com.skype.dev;
import com.skype.Skype;
//定义一个 SendChatMessage，用于向 Skype 用户发送聊天消息
public class SendChatMessage {
public static void main(String[] args) throws Exception {
        //对用户输入的参数进行判断，此参数需要在程序运行时指定
        if (args.length != 2) {
        //对输入参数的形式进行说明，参数需要输入要发送的另一个 Skype 用户的 ID 和消息
        内容
          System.out.println("说明：请输入程序启动运行的参数：参数为 'skype_id'
          'chat_message'，要呼叫的用户 ID 和要发送的消息，两个参数之间用空格分开。");
        //如果用户没有输入相应参数，或输入错误，程序直接返回
            return;
        }
        //将用户发送的消息内容和发送的时间打印出来，用来验证对比
        System.out.println("在" + Utils.showCurrentTime() + "时：向" + args[0]
        + "发送了内容为：" + args[1] + "的消息！");
        //调用 Skype 类的 chat()方法实现聊天功能，需要 Skype 用户 ID 和发送的消息内容
        作为参数
        Skype.chat(args[0]).send(args[1]);
    }
}
```

3. 程序验证和运行

SendChatMessage 类的运行，也需要输入相应的参数，参数的输入过程请参考 MakeCall 类的运行方法。需要注意的是，这里要输入两个参数，仍以 softwater 00756 用户为例，发送的消息为 Hello-This-is-test-SendChatMessge!，参数输入的形式如图 11.25 所示。

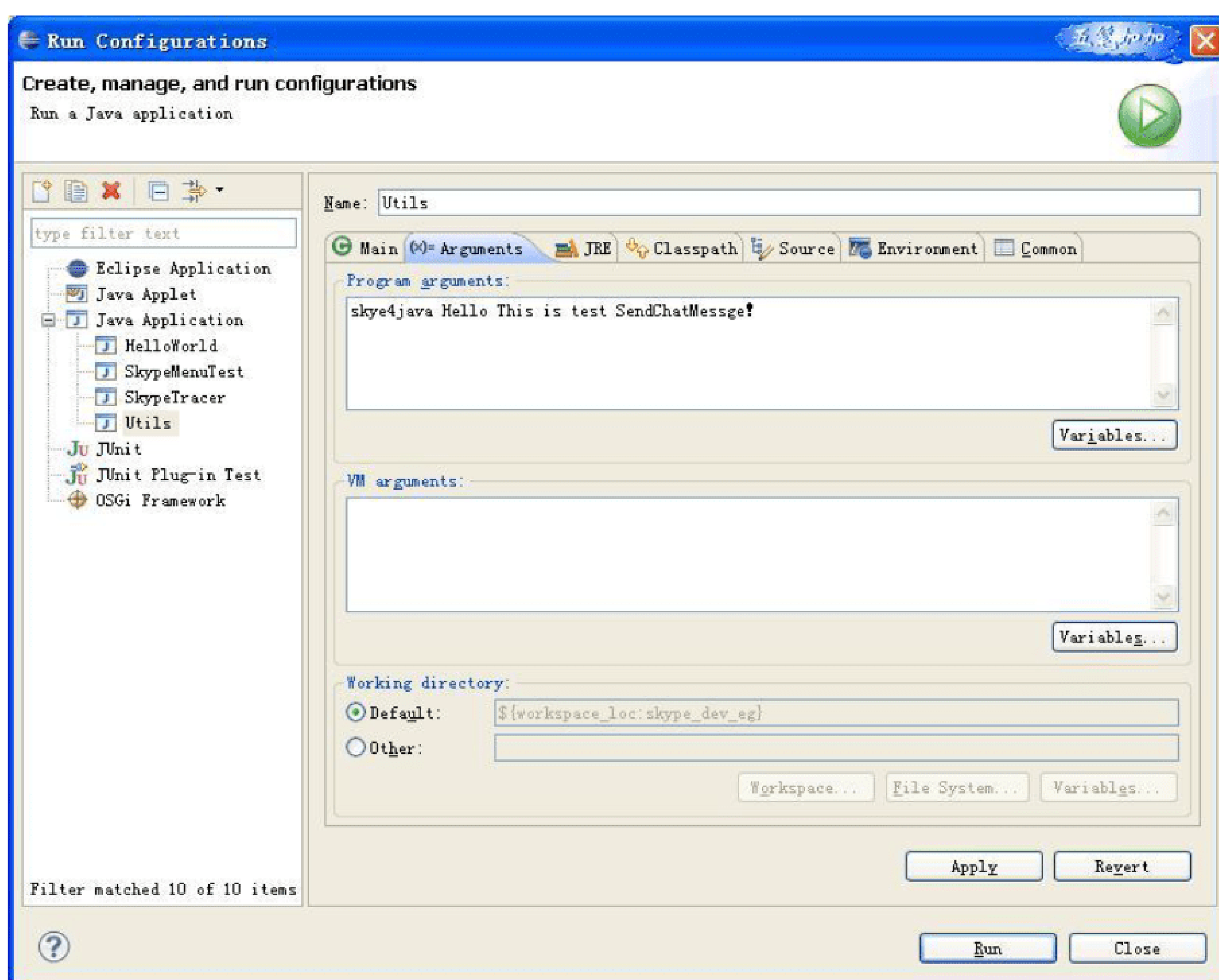


图 11.25　SendChatMessage 类运行时的参数输入情况

注意： 发送的消息参数，单词之间不能有空格，否则程序会误认为是多个参数，所以本例中的消息参数用短横线“-”来分隔单词。

在图 11.25 中，单击 Run 按钮运行，运行结果如图 11.26 所示。

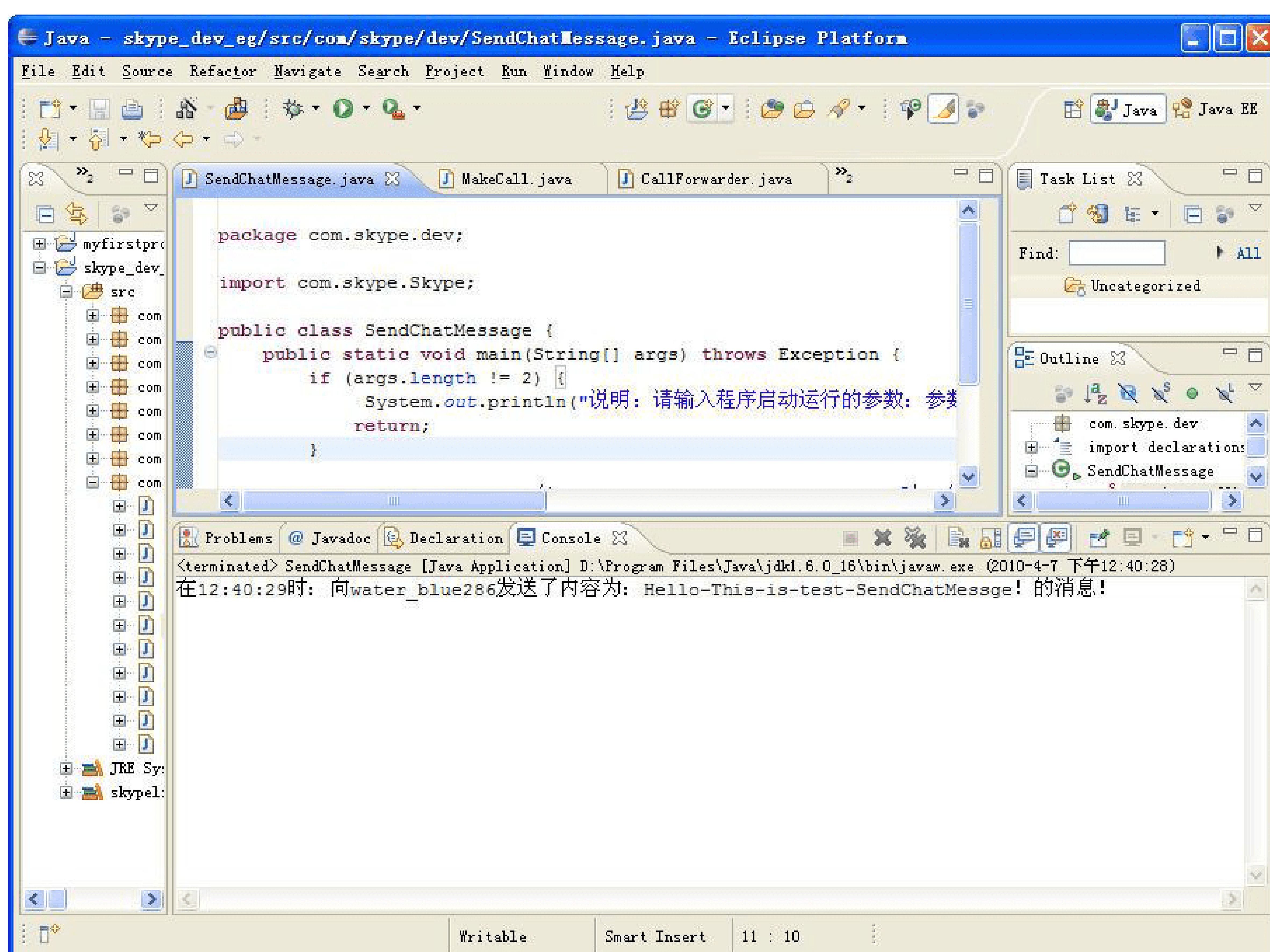


图 11.26　SendChatMessage 类运行后的控制台消息

从图 11.26 中可以看出，程序运行后，控制台打印的信息为“在 22:55:45 时：向 skye4java 发送了内容为 Hello-This-is-test-SendChatMessge！的消息！”。

这是应用程序在 water_blue286 一端发送消息的情况，再看看它所发送的好友 skye4java 一端的情况，如图 11.27 所示。



图 11.27　在 water_blue286 一端的消息显示图

从图 11.27 中可以清楚地看到，在 skye4java 客户端中，于 22：55 时，也收到了来自 water_blue 286 发送的“Hello-This-is-test-SendChatMessge！”的消息。

由此可以说明，通过应用程序实现了向 Skype 客户端发送聊天消息的功能。

11.6.3　实现自动应答

Skype 客户端的自动应答，简单地说，就是由应用程序来自动应答来自好友的聊天消息，就如同在使用 QQ 的时候，你突然有事走开了，如是启动了自动应答功能，此时每个给你发消息的好友，都由自动应答来回复消息。那么 Skype 客户端的自动应答要实现的就是这个功能。

1．实现机制

实现 Skype 客户端的自动应答的原理很简单，就是监听来自客户端所有的消息，然后针对每一条消息都进行自动回复。所以，首先要做的就是要添加一个监听器，然后在监听器收到消息后，进行自动回复。

Skype4Java 的开发包中，自动回复机制的监听器叫做 addChatMessageListener，它是 com.skype.Skype.java 类中的一个方法，关于 Skype 类中的 addChatMessageListener()方法，请参考随书光盘中的源代码，实现过程比较简单，读者可参考源码理解。

2．代码实现

Skype 开发包中的自动应答机制，在 Skype 插件中比较实用。有了这个机制用户可以定制一个插件，通过自动应答的功能自动回复一些特定的用户信息，也可以对一些垃圾信息、黑名单信息、陌生人信息等执行自动回复，这样能省去用户很多麻烦。

理解 addChatMessageListener()方法，也就理解了自动应答的原理，具体实现起来也很简单。在 skype_dev_eg 工程的 com.skype.dev 包中新建一个类，类名为 AutoAnswering.java，用于实现 Skype 客户端的自动应答功能。以下是自动应答方法的简要说明，程序源代码请参考随书所附光盘。

【源代码示例:\ch11\ch11_code\skype_dev_eg\src\com\skype\dev\ AutoAnswering.java】

```
  /**
   * AutoAnswering 类，用来实现 Skype 客户端的自动应答功能，对 Skype 用户而言，当用
   * 户因忙或不在线的时候，可以自动回复好友的消息，通过应用程序实现 Skype 客户端的自动
   * 应答。
   */
package com.skype.dev;
import com.skype.*;          //因为要用到 Skype 的相关方法，所以需要引入 Skype 类
public class AutoAnswering {    //定义一个 AutoAnswering 类，实现自动应答功能
    public static void main(String[] args) throws Exception {
        //调用 Skype 的 setDeamon()方法，设置为 false
        Skype.setDeamon (false);
        //添加一个聊天消息的监听器，用于监听来自好友的消息
        Skype.addChatMessageListener(new ChatMessageAdapter() {
            //通过一个匿名的内部类，来监听是否接收到消息
            public void chatMessageReceived(ChatMessage received) throws
            SkypeException {
               //对接收到的消息类型进行判断，如果是消息类型则自动回复，也可以指定其他
               类型
                if (received.getType().equals(ChatMessage.Type.SAID)) {
                 //当接收到消息时，向所发送消息的好友自动应答一条消息
                 System.out.println("自动回复消息的时间是：" +
                 Utils.showCurrentTime());
                 //通过 send()方法，将自动应答的消息发送出去
                 received.getSender().send("不好意思，我现在正在忙，稍后给您回复，
                 谢谢！");
                }
            }
        });
    }
}
```

3．程序的测试

运行 AutoAnswering 后，那么此应用程序所在的客户端就处于一种监听状态了，所有来自好友的消息都会由这个应用程序进行自动应答。

先启动 water_blue 286 一端应用程序的运行，此时发送 water_blue 286 的消息都被应用程序监听了，如图 11.28 所示，程序处于监听状态。

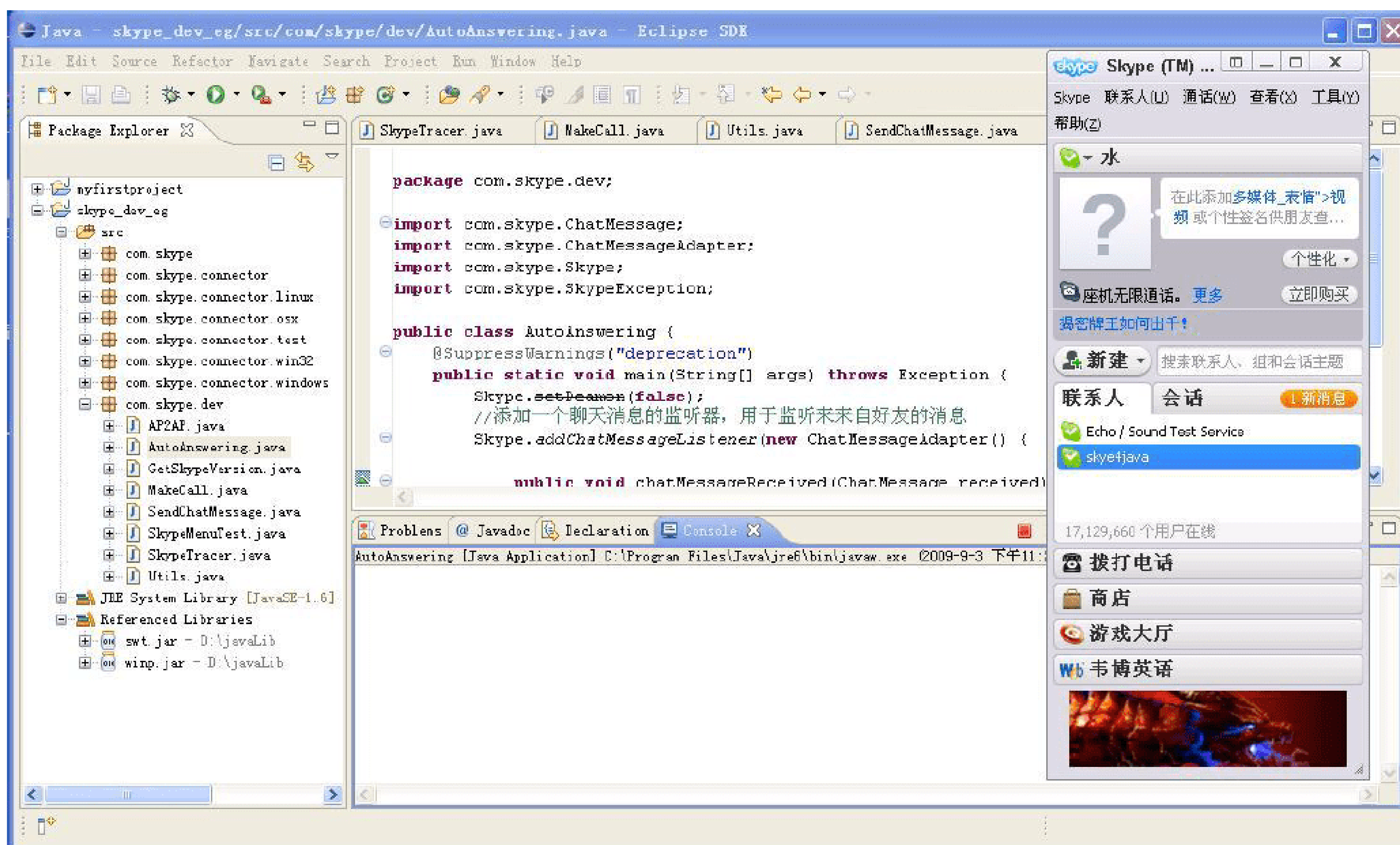


图 11.28　自动应答程序在 Skype 客户端处于监听状态

当自动应答程序处理监听状态时，在 water_blue 286 的好友，skype4java 一端向 water_blue 286 发送一条消息，如图 11.29 所示。



图 11.29　应用程序一端的好友发送的消息示意图

在图 11.29 中，skype4java 刚发完一条消息，即刻得到 water_blue286 的回复，回复的消息与应用程序中定制的回复消息是一样的，而且在 water_blue286 一端，Skype 没执行任何动作，应用程序的控制台上也打印了如下消息，如图 11.30 所示。

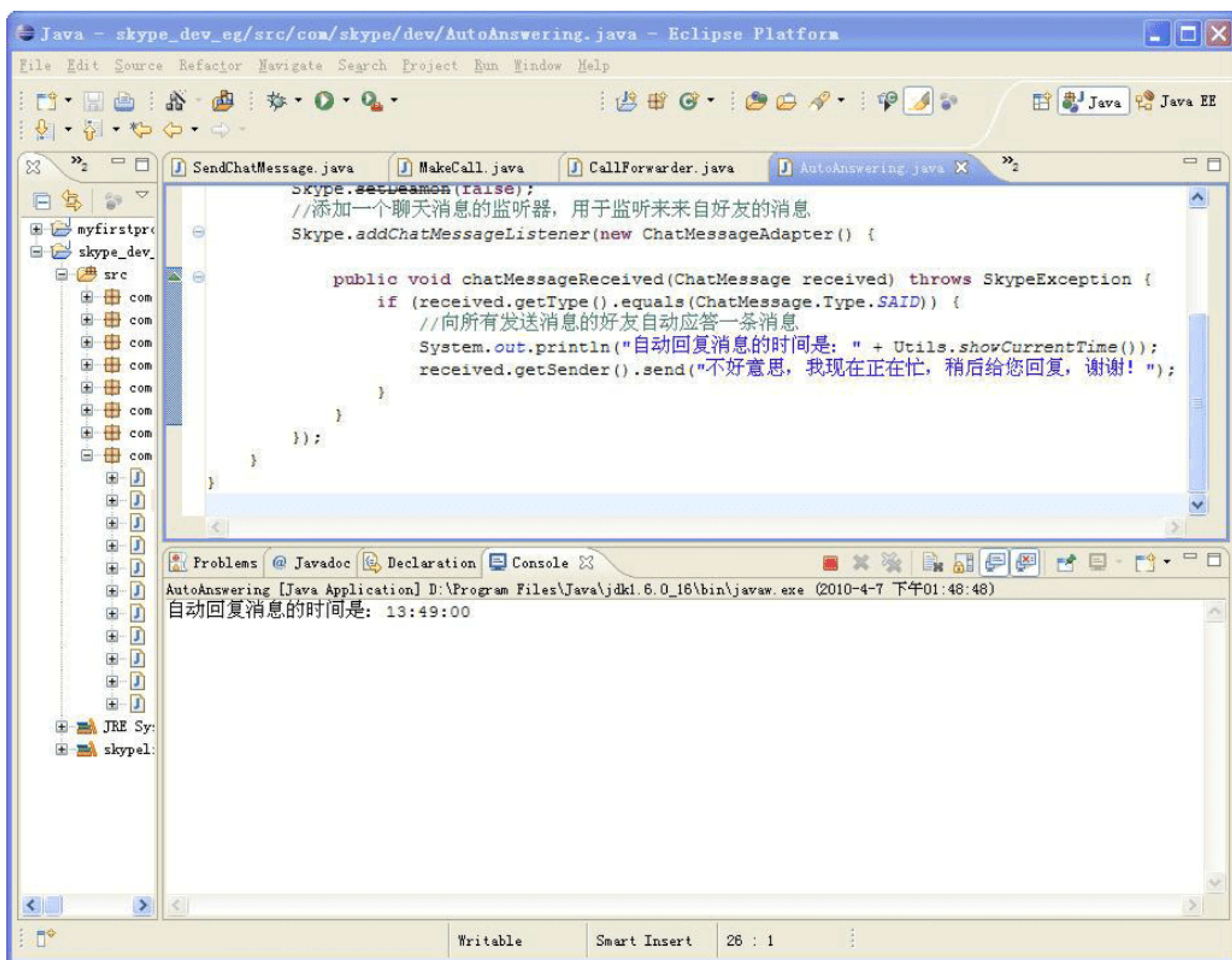


图 11.30　应用程序一端自动回复后的消息提示

从图 11.28 与图 11.29 中的时间显示上就可以看出，这条自动应答消息确实是由应用程序发出的，通过这个示例，就实现了 Skype 客户端自动应答的功能。

11.7　Skype4Java 其他的应用实例

11.6 节中讲解了 Skype 在呼叫、消息发送、自动应答等方面的基本应用开发，整个 Skype 的插件应用是非常广、非常多的，远不止这些，以上几个小例子只是为了让读者更好地理解插件应用程序的实现原理和基本方法。要实现真正的开发，还需要常常深入地演习和大量的实践

本节再介绍一些 Skype 的其他基本应用，这些应用将以程序源码的方式展示它们的实现方法，对其功能只作简单说明，不再详细地分析和验证运行结果。

注意： 读者最好自己输入这些代码，然后进行实际的操作和验证，这样能进一步加深理解。

11.7.1　呼叫转接

在 Skype 客户端中，当你的好友列表中有一个用户在向你呼叫的时候，运行呼叫转接的应用程序，就可以将来自好友的呼叫通过应用程序转接给其他的好友。因而，呼叫转接的功能就是，通过 Java 应用程序来截取 Skype 客户端中来自好友的电话呼叫，并将此呼叫转发给任意对象。这个功能在实际的 Skype 插件开发中，也有较广泛的应用。本例通过一个 CallForwarder.java 类，来实现 Skype 的呼叫转接功能，以下是关于 CallForwarder 类中主要方法的说明，程序的源代码请参考随书所附光盘。

【源代码示例：\ch11\ch11_code\skype_dev_eg\src\com\skype\dev\CallForwarder.java】

```
 /**
  * CallForwarder 类，主要用来实现呼叫转接的功能，基本的流程是，当 Skype 客户端接
  * 收到来自某一好友的呼叫时，此应用程序可以将此呼叫转发出去，转发的对象可以是另一个
  * Skype 用户，也可以是 PSTN 网络中的某一电话。在具体的转发过程中，还可以控制转发的开
  * 始时间，转发的持续时长等，有较广泛的实际应用。
  */
//定义一个 CallForwarder 类，用来实现 Skype 客户端的呼叫转发功能
public class CallForwarder {
  public static void main(String[] args) throws Exception {
        //调用 Skype 的 setDeamon()方法，并设置为 false
        Skype.setDeamon (false);
        //添加一个呼叫监听器，监听来自好友的呼叫
        Skype.addCallListener (new CallAdapter () {
            //通过一个嵌套的内部类，处理接收到的呼叫
            public void callReceived(Call receivedCall) throws
            SkypeException {
              //取出原有呼叫的转接规则，用一个 CallForwardingRule 对象数组表示
              CallForwardingRule [] oldRules = Skype.getProfile
              ().getAllCallForwardingRules ();
              //设置此次呼叫转接的方法，也就是将此次呼叫具体转接给谁
              Skype.getProfile().setAllCallForwardingRules(new
              CallForwardingRule[] { new CallForwardingRule(0,  30,
              "skype id") });
                //调用 receivedCall 的 forward()方法，将接到的呼叫转发出去
                receivedCall.forward ();
                try {
                    Thread.sleep(10000); //让进程暂停 10 秒钟，也就是持续呼叫 10 秒
                } catch (InterruptedException e) {
              }
                //将转发后的信息打印输出，用于验证对比
                System.out.println("此呼叫已经转发，转发的用户为：" + "skype_id");
                //进行复位操作，还原现场，将转接规则设置成原来的方式
                Skype.getProfile ().setAllCallForwardingRules (oldRules);
            }
        });
    }
}
```

以上的这段代码是一个呼叫转发的核心，在这个方法中涉及对其他类的引用、其他方法的调用，读者要结合源码进行理解。

还需要强调一点的是，关于 CallForwardingRule()方法的使用，此方法主要用来定义呼叫转发的规则，有了这个规则，应用程序就可以根据规则进行定制式的转发。CallForwardingRule()方法中的转发规则通过参数来定义，这些参数具体定义了呼叫转接的对象、转发开始的时间，转发后呼叫持续时间等，关于 CallForwardingRule()方法的原型说明如下：

```
/**
* 构建一个呼叫转接的规则
* param newStartSecond，呼叫发出以后，从哪个时间开始转接
* param newEndSecond，转接呼叫开始后，持续多长时间呼叫接束，也就说是持续呼叫响铃的
* 时间 param newTarget，要转接的目标，是一个 SkypeID 的对象参数，可以是一个 Skype 用
* 户的 ID 也可以是 PSTN 网络里的一个电话号码。
 */
public CallForwardingRule(int newStartSecond, int newEndSecond, String
newTarget) {
            this.startSecond = newStartSecond;   //记录开始时间
            this.endSecond = newEndSecond;       //记录结束时间
            if (newTarget.startsWith("+")) {     //对转发的目标进行判断
                newTarget = newTarget.replaceAll("-", "");
                                                 //对新的呼入目标进行处理
            }
            this.target = newTarget;             //将新目标赋值给转发的对象
        }
```

在这个测试中，最好有 3 个 Skype 账号，假设这 3 个 Skyep 账号分别为 UserA、UserB、UserC，在 UserA 的一端运行 CallForwarder.java 程序，程序代码中转接的目标对象设置为 UserB。也就是将程序代码中 skype_id 换成 UserB，这样所有到达 UserA 的呼叫都会被转接到 UserB 上。

此时，在 UserC 的一端呼叫 UserA 的时候，UserA 一端的应用程序会打印一条说明消息："此呼叫已经转发，转发的用户为 UserB"，这样，在 UserB 的一端就会接到 UserC 的呼叫，并且从转接的那一刻开始持续响铃 30 秒。

通过这种方式就实现了 Skype 的呼叫转接功能。

11.7.2　呼叫终止

呼叫终止的意思是，当在 Skype 客户端运行呼叫终止程序的时候，所有的此 Skype 客户端的呼叫都将被终止，这种终止包括两个方面的呼叫，一个是主叫，另一个是被叫。

当呼叫终止程序所在的 Skype 客户端向外呼叫时，是主叫终止，而由外部发向此客户端的呼叫，是被叫终止。所以呼叫终止的功能就是，利用 Java 程序来结束所有的 Skype 客户端的呼叫。

在 skype_dev_eg 工程的 com.skype.dev 包中新建一个类，类名为 ShutdownCall.java，用于终止在应用程序所在 Skype 客户端的所有呼叫。以下是关于 ShutdownCall 类的主要方法说明，此类的源代码请参考随书所附光盘。

【源代码示例：\ch11\ch11_code\skype_dev_eg\src\com\skype\dev\ShutdownCall.java】

```
 /**
  * ShutdownCall 类，主要功能是终止 Skype 客户端的所有呼叫，包括由 Skype 客户端发
  * 的主叫和发向此 Skype 客户端的被叫。对于用户而言，一旦运行 ShutdownCall 类，所有与
  * 此 Skype 客户端有关的呼叫，都将结束，具体实现如下：
  */
package com.skype.dev;
import com.skype.Call;
import com.skype.CallAdapter;
import com.skype.Skype;
import com.skype.SkypeException;
//定义一个 ShutdownCall 类，用于实现呼叫终止的功能
public class ShutdownCall {
  public static void main(String[] args) throws Exception {
        //调用 Skype 类的 setDeamon()方法，并设置为 false
        Skype.setDeamon (false);
        //通过 Skype 的 addCallListener()方法，载入呼叫监听器，对所有呼叫行为进行
        监听
        Skype.addCallListener (new CallAdapter () {
            //通过 callMaked()方法，处理主叫行为
            public void callMaked(Call makedCall) throws SkypeException {
              System.out.println("主叫被终止！");  ///打印主叫终止信息
              makedCall.finish();     //通过 makedCall 的 finish()方法来终止主叫
            }
            //通过 callReceived()方法，处理被叫行为
            public void callReceived(Call receivedCall) throws
            SkypeException {
              System.out.println("被叫被终止！");   //打印主叫终止信息
              receivedCall.finish();
                                   //通过 receivedCall 的 finish()方法来终止被叫
            }
        });
    }
}
```

这段程序中，主要调用了与处理呼叫有关的两个方法，一个是处理主叫终止的命令，为 makedCall.finish()，另一个是处理被叫终止命令，为 receivedCall.finish()，具体的方法含义请读者结合源代码和相关文档来理解。

11.7.3　综合性的 Skype 应用实例

这个综合的小应用，就是模拟 Skype 的工作过程，实现连接、呼叫、发送消息等基本功能。它是一个 AP2AP 的应用，也就是在两个 Skype 客户端同时运行此应用程序，然后这两个应用程序之间来模拟实现 Skype 的相应功能。

1. 代码实现

这个应用程序是基于 SWT 开发的，所以需要再导入一个包，具体方法是将一个 swing-layout-1.0.jar 包复制到 javaLib 目录下，再按照导入 Jar 的方式将此包载入到工程的

构建路径下。然后在 skype_dev_eg 工程的 com.skype.dev 包中新建一个类，类名为 SkypeAppTest，建好后的 skype_dev_eg 工程目录结构如图 11.31 所示。

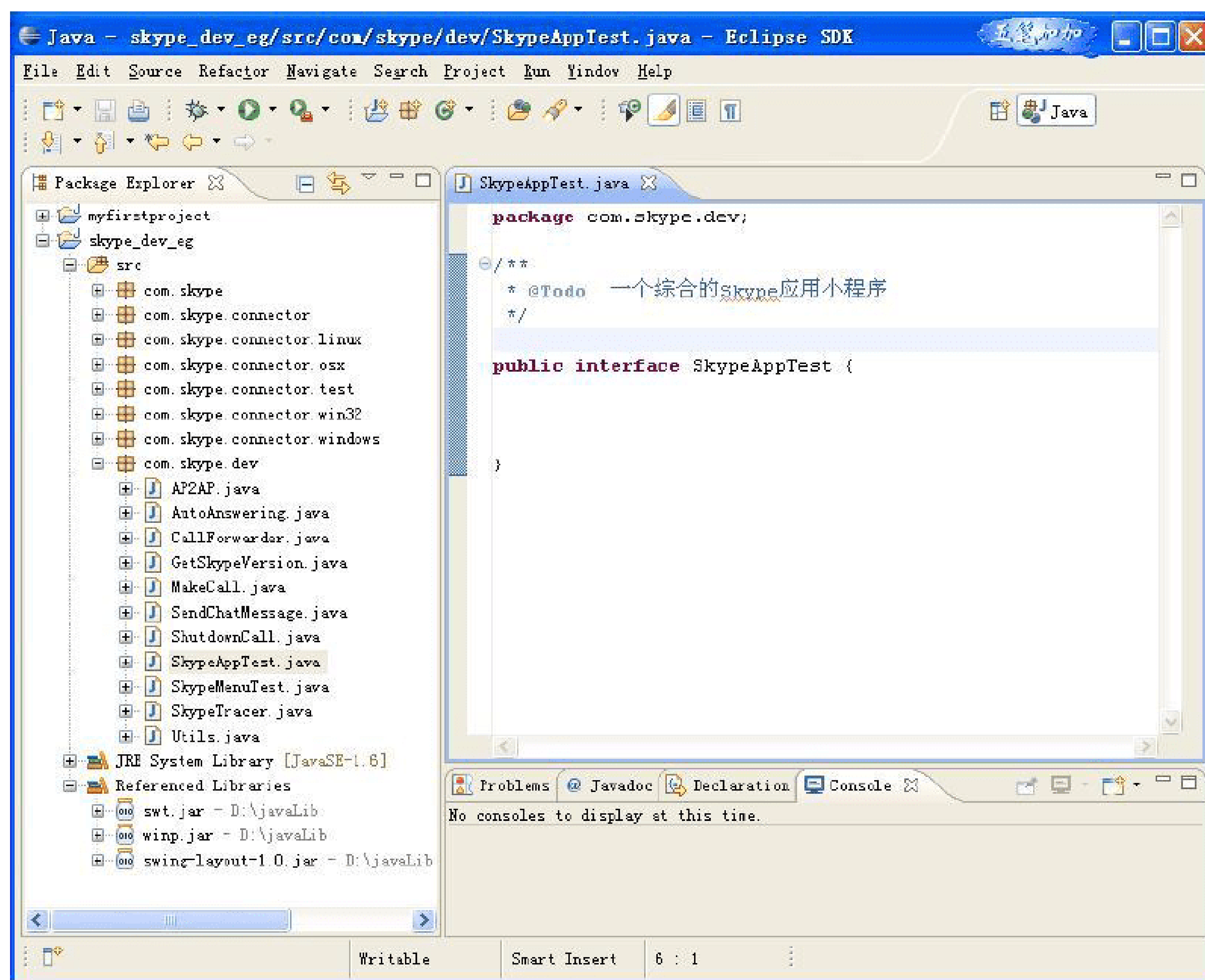


图 11.31　skype_dev_eg 工程目录结构图

注意： swing-layout-1.0.jar 包是用于 Java 图形界面开发时布局管理用的，在网络上可以下载，在本书的随书光盘中也都提供了这些 Jar 包。

SkypeAppTest 类是一个综合的应用，包括界面的显示、应用程序之间的连接、产生一个呼叫、发送聊天消息等基本功能。下面就对这些功能的主要实现方法进行说明，其他与之相关的方法和详细的源代码请参考随书光盘。

【源代码示例：\ch11\ch11_code\skype_dev_eg\src\com\skype\dev\SkypeAppTest.java】

要开发一个完整的应用程序，首先要完成类的定义、包的引入、变量的初始化等工作。以下就是 SkypeAppTest 类的包声明和相关变量的定义。

```
package com.skype.dev;                  //声明 SkypeAppTest 所在的包路径
/**
 * SkypeAppTest 类，一个综合性的 Skype 应用小程序，将 Skype 应用程序关于连接、呼叫、
 * 发送消息等功能集成到一个统一的界面中，以集中地展示并应用这些功能，重点是学习如何整合
 * 开发一个综合性的 Skype 插件应用。
 */
//以下是引入 Skype 应用开发所要的包、对象
import com.skype.Application;           //引入 Application 对象
import com.skype.Call;                  //引入 Call 对象，完成呼叫有关的操作
import com.skype.ContactList;           //引入 ContactList 对象，与联系人列表有关
import com.skype.Friend;                //引入 Friend 对象，与好友操作有关
import com.skype.Skype;                 //引入 Skype 对象，完成呼叫、发消息等操作
import com.skype.SkypeException;        //引入 SkypeException 对象，与异常操作有关
import com.skype.Stream;                //引入 Stream 对象，与数据流交互有关
```

```
import com.skype.connector.Connector;  //引入 Connector 对象，与连接操作有关
import com.skype.connector.ConnectorException;
                                     //引入 ConnectorException，处理连接异常
 /**
 * 以下是关于对 SkypeAppTest 类进行的声明和定义，整个类继承了 javax.swing.JFrame 类，
 *同时也实现了 com.skype.StreamListener 和 com.skype. ApplicationListener 两个
 *接口。
*/
public class SkypeAppTest extends javax.swing.JFrame implements .
ApplicationListener, com.skype.StreamListener {
    /**
    *以下是关于整个界面布局的定义， 此应用程序的界面中，包括 JButton、JTextField、
    *JTextArea JScrollPane 等基本的布局元素，这些元素具体在 javax.swing 包中。
    */
    private javax.swing.JButton callButton;         //定义一个呼叫的按钮
    private javax.swing.JButton connectButton;      //定义一个建立连接的按钮
    private javax.swing.JTextField friendField;     //定义一个文本输入框
    private javax.swing.JScrollPane jScrollPane1;   //定义一个滚动面板
    private javax.swing.JTextArea jTextArea1;       //定义一个文本域
    private javax.swing.JTextField messageField;    //定义一个文本框，显示消息
    private javax.swing.JButton sendButton;         //定义一个发送消息的按钮
    /**
     *以下是关于应用程序中用到的一些类的声明，包括 Application 类、Stream 类、Boolean
     * 型的 connected 变量，Call 类等。
     */
    Application myApp;                 //定义一个 Application 实例 myApp
    Stream myStream;                   //定义一个 Stream 实例 myStream
    Boolean connected;                 //定义 Boolean 型的 connected，判断是否连接
    Call myCall;                       //定义一个 Call 实例 myCall
```

完成了以下基本的类的定义、变量的声明，下面就是对主体的方法实现。SkypeAppTest 类中的功能，由一个构造方法来完成，这样，创建一个 SkypeAppTest 实例，就实现了一个基于 SkypeAppTest 的应用。SkypeAppTest 的构造方法实现如下：

```
   /**
   * SkypeAppTest 类的构造方法，与类名相同，封装了 SkypeAppTest 应用程序所定义的全
   * 部功能在构造方法中统一调用各功能方法，实现方法如下：
   */
//声明一个 public 的 SkypeAppTest 构造方法
public SkypeAppTest()  {
       //初始化应用程序的组件信息，用于完成界面的显示、初始化等基本功能
       initComponents ();
       try {
           // 打印了 Skype 的版本信息，测试系统能否正常工作
           System.out.println("当前的 Skype 版本为： " + Skype.getVersion());
           // 通过 Connector 实例的 executeWithId()方法，执行一个 Skype API 的相
           关命令
           try {
               Connector.getInstance ().executeWithId ("MESSAGE seanmwhite
               Hi from UI","CHATMESSAGE");
           //捕获连接异常并处理
           } catch (ConnectorException ex) {
               ex.printStackTrace ();
```

```
            }
            //创建一个 Skype 应用，传入一个参数进行命名，这里取名为 skypeapp
            myApp = Skype.addApplication("skypeapp");
            //为这个名为 skypeapp 的应用添加一个监听器
            myApp.addApplicationListener (this);
            //用一个 Boolen 型的 connected 标识连接的情况，避免进行多次重复连接
            connected = false;
            //捕获并处理 SkypeException
        } catch (SkypeException ex) {
            ex.printStackTrace ();
        }
    }
```

以上的构造方法很概括地封装了 SkypeAppTest 中的基本功能方法，这些功能方法由构造方法统一调用，它们主要完成连接和建立与释放、处理数据流、产生呼叫、发送聊天消息等功能。下面分别对这些功能方法进行说明。

在 SkypeAppTest 中，要实现两个应用程序之间的通信，首先要完成连接的功能，同时在连接结束后，还要处理连接释放的动作，下面就是连接的建立与释放的方法。

```
/**
 *以下的两个方法为 connected()和 disconnected()，它们分别完成连接的建立与释放的功
 *能，都接收一个 Stream 对象作为参数，具体的实现如下
 */
  //建立连接的方法，名为 connected，传入 Stream 对象作为参数
  public void connected(Stream stream){
      //对传入的 Stream 对象，添加一个 addStreamListener 的监听器
      stream.addStreamListener (this);
      //设置 Boolean 型的 connected 的值为 true，说明已连接
      connected = true;
      //将传入的 Stream 的一个实例，赋值给 myStream 变量
      myStream = stream;
   }
  //释放连接的方法，名为 disconnected，传入 Stream 对象作为参数
   public void disconnected(Stream stream) {
      //对传入的 Stream 对象，通过 removeStreamListener()方法，移除其监听器
     stream.removeStreamListener (this);
     //设置 Boolean 型的 connected 的值为 false，说明当前处于非连接状态
     connected = false;
     //将 myStream 变量的值赋为 null
     myStream = null;
  }
```

在 SkypeAppTest 的应用中，当两个应用程序建立好连接以后，还需要处理数据的接收问题，以下就是数据接收方法的具体实现。

```
/**
 *SkypeAppTest 应用程序中，包含两类数据的接收，所以定义了两个接收数据的方法，这两个
 *方法分别为 datagramReceived()和 textReceived()，根据不同的数据类型采用不同的
 * 接方法，具体的实现过程也很简单。
 */
  //处理接收到数据的两个方法，需要传入 String 类型的数据作为参数
  public void datagramReceived(String receivedDatagram) {
     //将接收到的数据设置到文本域中
     jTextArea1.setText(receivedDatagram);
```

```
    }
    public void textReceived(jString receivedText) {
        //将接收到的数据设置到文本域中
        jTextArea1.setText(receivedText);
    }
```

完成了以上的方法后，下面要做的就是在界面的各个按钮中加入监听动作，当不同的按钮按下时，触发此动作，执行相应的功能。这样，整个应用程序的功能就集中统一了。

在所有的按钮动作中，连接按钮会最先被执行，此按钮触发连接动作，用来完成两个应用程序之间的连接。

```
    /**
     * connectButtonMouseClicked()方法，用来处理“连接”按钮触发后的相应动作，执行
     * 连接功能当用户执行 SkypeAppTest 时，单击连接按钮，程序会向所连接的用户发出连接
     * 请求，然后两个应用程序之间就会建立一个连接的通道，接着就可以执行其他的相关操作。
     * 具体实现如下：
     */
//connectButtonMouseClicked()方法，用来实现连接功能，需要传入 MouseEvent 对象作
为参数
private void connectButtonMouseClicked(java.awt.event.MouseEvent evt)
        //首先会对当前的连接状态进行判断，如果是已经连接的，就不再执行连接动作
        if (!connected)
            //如果当前没有连接，则执行以下的连接过程
            try {
                //从用户的联系人列表，取出用户的好友列表
                ContactList myFriends = Skype.getContactList ();
                //指定一个需要连接的 Friend 对象，myFriend
                Friend myFriend = myFriends.getFriend (friendField.getText
                ());
                //将与之连接的 Friend 对象名称打印输出
                System.out.println (myFriend.getFullName ());
                //直接调用 Application 实例的 connect()方法，与传入 Friend 对象建立
                连接
                myApp.connect (myFriend);
            //处理 SkypeException 异常
            } catch (SkypeException ex) {
                ex.printStackTrace();
            }
    }
```

以上就是在“连接”按钮上执行建立连接功能的方法。连接建立完成以后，就是执行呼叫、发送聊天消息等基本操作，以下就是针对“呼叫”按钮和“发送消息”按钮的监听方法。

```
    /**
     * 本段代码，主要完成“呼叫”按钮和“发送消息”按钮的动作，以实现呼叫和发送消息的基
     * 本功能，呼叫按钮执行动作由 callButtonMouseClicked 方法完成，传入 MouseEvent
     * 对象作为参数，当用户单击此按钮时，SkypeAppTest 应用程序会向指定的用户产生一个呼
     * 叫，而发送消息按钮的执行动作由 sendButtonMouseClicked 方法完成，也需要传入
     * MouseEvent 作为参数，当用户单击发送消息按钮时，SkypeAppTest 应用程序会向指定的
     * 用户发送一条消息。具体实过程如下：
     */
    //定义一个 private 的 callButtonMouseClicked，用来完成“呼叫”按钮的动作
    private void callButtonMouseClicked(java.awt.event.MouseEvent evt) {
```

```
        try {
           //调用 Skype 的 call()方法，根据传入的 Friend 信息，产生一个对此 Friend
           的呼叫
           myCall = Skype.call(friendField.getText());
         //捕获并处理 SkypeException
        } catch (SkypeException ex) {
           ex.printStackTrace ();
        }
    }
    //定义一个 private 的 sendButtonMouseClicked，用来完成“发送消息”按钮的动作
    private void sendButtonMouseClicked(java.awt.event.MouseEvent evt) {
      //首先对当前的数据传输通道进行判断，当数据传输流存在且不为 null 时，执行发送动作
          if (myStream != null)
           try {
              //直接通过 myStream 对象的 send()方法，将用户输入的消息发送出去
               myStream.send (messageField.getText ());
             //捕获并处理 SkypeException
           } catch (SkypeException ex) {
               ex.printStackTrace ();
           }
    }
```

完成了以上的方法以后，整个 SkypeAppTest 应用的基本功能也就全部完成了，剩下的工作就是进行应用程序的界面开发。也就是完成构造方法中的 initComponents ();方法，此方法主要用来设置按钮名称、设置界面布置、添加按钮的监听动作、显示会话信息等，主要涉及的是 Java 界面编程知识，读者可参考源代码进行学习。

注意： 本例中的界面控制和布局是由 Javax.swing 包中的相关类实现的，在编程过程中需要引入相应的 Jar 包。

这样，整个 SkypeAppTest 的综合应用程序就开发完成了，在本实例中，没有什么难理解的地方，主要是在前面内容中讲过的几个应用方法基础上的整合与实现。相关代码的含义都给了明确的注释，读者要在理解的基础上牢牢掌握 Skype 综合应用的一般开发方法。



图 11.32　SkypeAppTest 类运行示意图

2. 程序的运行与测试

这是一个 AP2AP 的应用程序，所以需要在两个 Skype 客户端同时运行此程序，而且连接的时候需要输入对方的 Skype ID 号，程序启动后如图 11.32 所示。

如果在另一端也启动了 SkypeAppTest.java 的运行，那么，在输入相互的 Skype ID 号再进行连接后，两个应用程序就可以进行连接、呼叫、发送消息等操作了。图 11.33 与图 11.34 展示了这两个应用程序之间通信的情况。

由图 11.33 与图 11.34 的对比可以发现，这两个应用程序之间可以进行消息的发送，也就是说，通过此应用程序，可以不在 Skype 客户端中而进行 Skype 网络中的消息交互。

至于连接、呼叫等功能，经测试也没有任何问题。



图 11.33　water_blut 286 一端应用程序运行的情况



图 11.34　Skye4Java 一端应用程序运行的情况

Skype 基本应用的开发就讲到这里了。本书所讲的并不是带领读者开发一个实在的 Skype 插件或应用程序，而是通过对 Skype 应用开发的原理、思想、基本方法的讲解，让读者明白开发 Skype 应用的基本流程和实现方法，掌握 Skype4Java 开发包的使用，理解 Skype API 的核心方法和思想。学完本章，希望读者能在此基础上设计和开发出更多的 Skype 高级应用。

注意：本章所有的源代码都在随书光盘中，请读者根据对应的章节找到相应的源码对照着参考开发。

11.8　本章小结

本章重点讲解了一整套的 Skype 应用开发过程，用 Skype 开发包讲起，分析了 Skype API 的层次结构，接着讲解了 Skype4Java 包的实现原理、基本架构及进行快速开发的方法。然后以 Skype4Java 包为基础，讲解了 Skype4Java 包的结构、重要的对象、方法等，最后以实例的方式细致的讲解了 Skype 基本功能开发的整个过程。

Skype 作为 P2P 领域最典型的即时通信系统，具有非常大的商业价值和发展潜力，学习 Skype 应用的开发，不仅能进一步理解基于 P2P 的 Skype 原理和通信流程，也能为自己将来的发展打下良好的基础。本书中的实例希望读者能对照着书中的讲解一步步操作下去，相信你在编程实践中会得到更大的收获。

第 12 章　基于 P2P 的即时通信系统的开发与实现

即时通信是目前互联网最为流行的通信方式，各种各样的即时通信软件也层出不穷。随着互联网的发展以及 3G 网络的到来，即时通信的应用将更加广泛，因此，研究并学习即时通信系统的开发技术对未来发展很有意义。

在本书的第 8 章，已经详细地讲解了即时通信的相关知识，对基于 P2P 技术的即时通信系统也作了重点的阐述，在理论层面上让读者对即时通信的整个知识体系有了系统的理解。本章将从实践应用出发，以案例的形式，带领读者开发一个简单而又实用的基于 P2P 的即时通信系统，使读者进一步掌握 P2P 技术的实践与应用开发的能力。

本章的重要知识点如下。

- 系统的规划：需要了解系统设计的基本目录、组织结构及功能结构等，对任何系统开发而言这是最基本的要求，只有规划好了，后面开发中才会做到有的放矢。
- 系统需求分析：了解系统的一般需求，掌握系统在通信、文件传输这两大功能上的基本要求。
- 系统关键技术及实现机制：要重点掌握在系统开发中所用到的关键技术，掌握系统的实现机制。
- 类的组织与设计：能对系统进行模块化的划分和进行面向对象的设计，能够规划每个类的基本功能，能对系统的全局关系有全面的了解。
- 系统的开发与编程实现：这是本系统开发的重要实践环境，掌握 Java 工程创建的基本方法，能通过程序源码来实现具体的功能；能有效地组织各个类之间的关系；能够调试和检测开发过程中出现的问题和错误。
- 系统的测试：能将系统正常地运行起来，能根据系统设计的目标和需求来检验相应的功能。
- 系统的打包与发布：要掌握系统打包的方法，能够按工程开发的要求生成完整的发布程序。

12.1　系 统 规 划

系统规划主要用来描述整个 P2P 即时通信系统的设计目标、组织结构及功能结构等，从整体上把握整个系统开发及设计的基本要求。

12.1.1　系统的设计目标

设计一个用于互联网的基于 P2P 即时通信工具，使其具有 P2P 的特性同时又有一般即时通信工具都具有的功能，其设计目标主要包括以下几个方面：

- 其通信的交互过程是架构在 P2P 技术之上的，没有中心服务器，也就是说，通信的实质是点对点进行的，无须服务器中转。
- 每一个用户都可以新建一个自己的频道，或是加入当前网络中已有的频道，在同一频道的任意两个 Peer 之间可以进行消息、文件、共享数据等交互。
- 当一个新的 Peer 加入到网络中时，可以新建一个频道，也可以搜索当前的频道信息，选择一个已有的频道加入。
- 同一频道的所有 Peer 之间可以共享文件，发送、接收文件，交互文件信息等。
- 同一频道的任意两个 Peer 之间可以进行私人会话。
- 能实现即时通信的其他功能，如界面展示、消息提示、好友列表、登录认证等。

注意： 本章只为演示一下基本的基于 P2P 技术的即时通信系统，所以整个系统的设计目标要求很低，重点体现在 P2P 通信、Peer 间简单的数据交互和文件共享上，功能很简单，无法与真正的即时通信系统相比，因而本系统仅供学习及练习使用。

12.1.2　系统总体组织结构

根据以上的系统目标，要设计一个满足以上要求的即时通信系统，在系统的整体设计结构上应该由以下几个层次组成。

- 显示模块：UI 操作界面；
- 功能模块：用于完成即时通信、文件传输及目录共享的基本功能；
- 消息模块：所有用于在网络上进行交互的各种数据信息的处理；
- 网络模块：实现 P2P 的网络机制。

这 4 个模块之间的结构关系如图 12.1 所示。

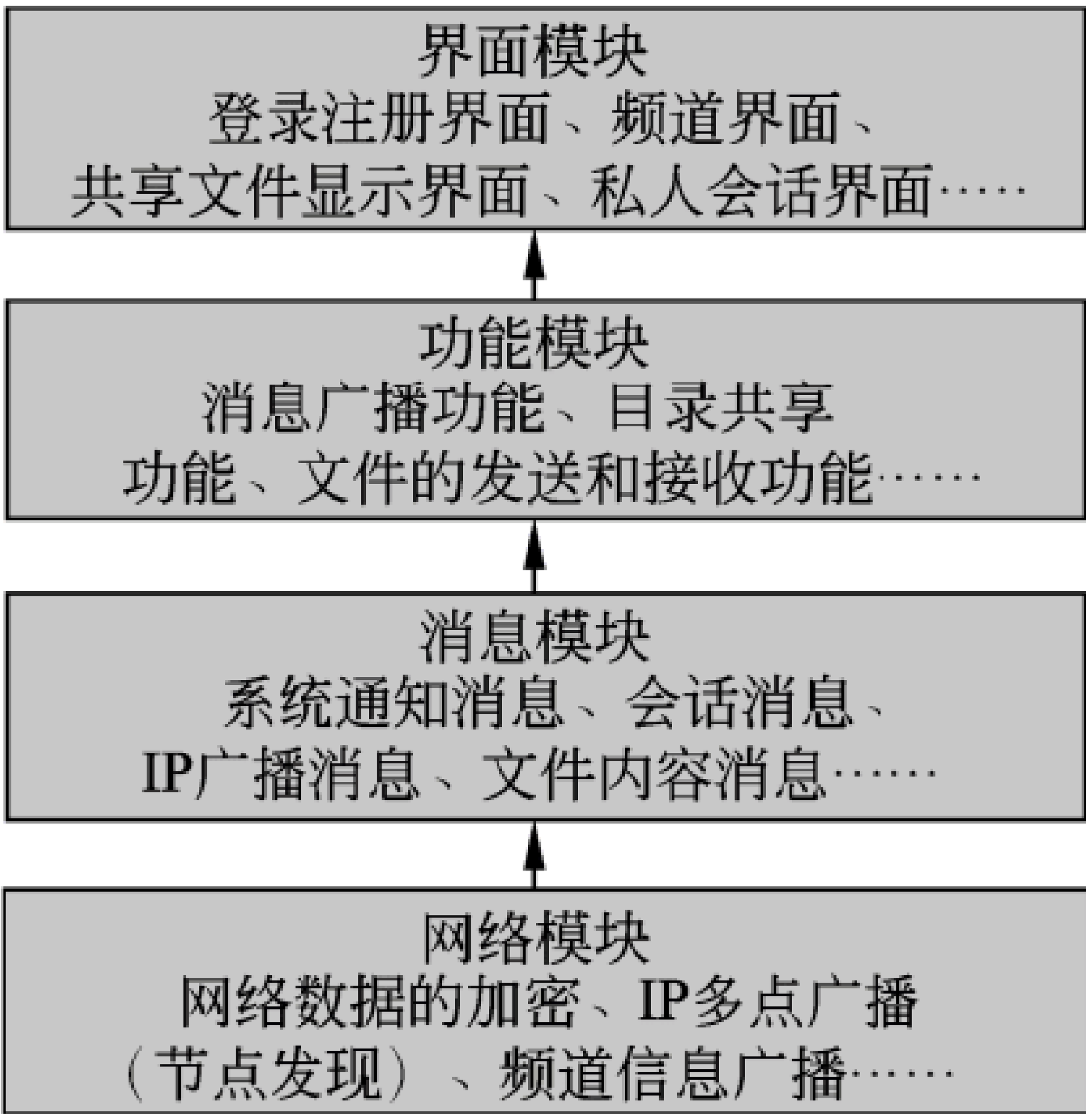


图 12.1　P2P 即时通信系统的基本组织结构图

如图 12.1 所示，P2P 即时通信系统的整体结构分 4 个层次，这 4 个层次之间是层层递进、相互联系的。在底层就是网络层，主要用于实现 P2P 的机制，在网络层的基础上通过消息层来实现 Peer 结点之间所有消息的发送和交互，然后用功能层来实现即时通信、文件传输与共享等基本功能。最后，通过界面层进行统一的封装，形成一个可视的、交互性好的完整系统。

使用这种结构，可以让系统的各个部分之间形成一种既独立又统一的依赖关系，这样，结构设计就与设计目标达到了有效的统一，便于模块化编程。

12.1.3　系统功能结构

根据设计目标的要求，P2P 即时通信系统的功能应该由以下几个基本功能模块组成。

- 登录认证与注册创建的功能；
- Peer 间的即时通信功能；
- 文件交互功能；
- 文件共享功能；
- 其他的功能。

对照着以上这几个基本功能，下面再详细地描述一下这些功能在系统运行中主要扮演什么角色，要完成哪些任务。

1．登录认证与注册创建的功能

在程序启动后有一个登录与注册的界面，在这个界面中，用户可以有两种选择。

- 第一个选择：可以根据当前的频道列表显示的信息，选择当前网络中已有的一个 P2P 频道，输入此频道的认证密码后，加入该频道。
- 第二个选择：用户可以创建一个新的 P2P 频道，输入用户名称、频道的名称和认证密码后，一个新的频道就创建好了。

注意： 这里所说的频道是一个模拟的 P2P 网络，用在即时通信系统中类似于聊天室的功能。以上的两种选择等价于，第一种选择，当你要加入一个聊天室时，只需选择一个已有聊天室的名字，然后输入你的用户名和聊天室的认证密码，就可以加入此聊天室了。而第二种选择，可以创建一个新聊天室，自定义聊天室的名字和密码，这样，别人就可以加入这个聊天室了。

登录一个 P2P 频道或创建一个新的 P2P 频道，是本系统中 Peer 结点发现的基础，这些都是基于底层的 IP 多播实现，在后文会有详细的讲解。

2．Peer间的即时通信功能

Peer 间的即时通信，确切地说应该是同一 P2P 频道内的所有 Peer 间的通信，这个通信过程包括两个方面：

- 一方面：所有 Peer 间的广播通信，类似于多人聊天室，某一 Peer 发布的消息，在同一 P2P 频道内的其他所有 Peer 到能接收到此消息。
- 另一方面：两个 Peer 间的私有通信，通信过程是在两个 Peer 之间进行的，其他 Peer

不参与这一过程，通信的消息内容对其他 Peer 不可见。

3．文件交互功能

文件交互功能包括文件的发送和接收，Peer 一端可以向同一频道内的所有 Peer 发送文件，也可接收来自同一频道内的其他任何 Peer 发送的文件。也就是说 Peer 可以在频道内广播发送的文件，这样，Peer 之间可以进行任意的文件和数据的交互。

4．文件共享功能

文件共享功能，指的是 Peer 一端，可以将本地的文件目录共享出去，这样，在这个 P2P 频道内的其他所有 Peer 都可以访问这个共享目录，也可以从 Peer 列表中查看其他 Peer 共享出来的目录。在查看目录的同时，也能查看共享的文件列表、文件内容等。

5．其他功能

除了以上的功能外，系统还提供一些其他必需的功能，如清屏操作的功能，就是将消息内容界面清空。系统服务消息功能，如系统提供的一些错误的提示消息、系统运行的状态消息、Peer 结点的通知消息等，这些也都是需要具体实现的功能。

以上描述的是整个系统的整体功能，可用一个基本的功能结构图来简单直观地表示，功能结构如图 12.2 所示。

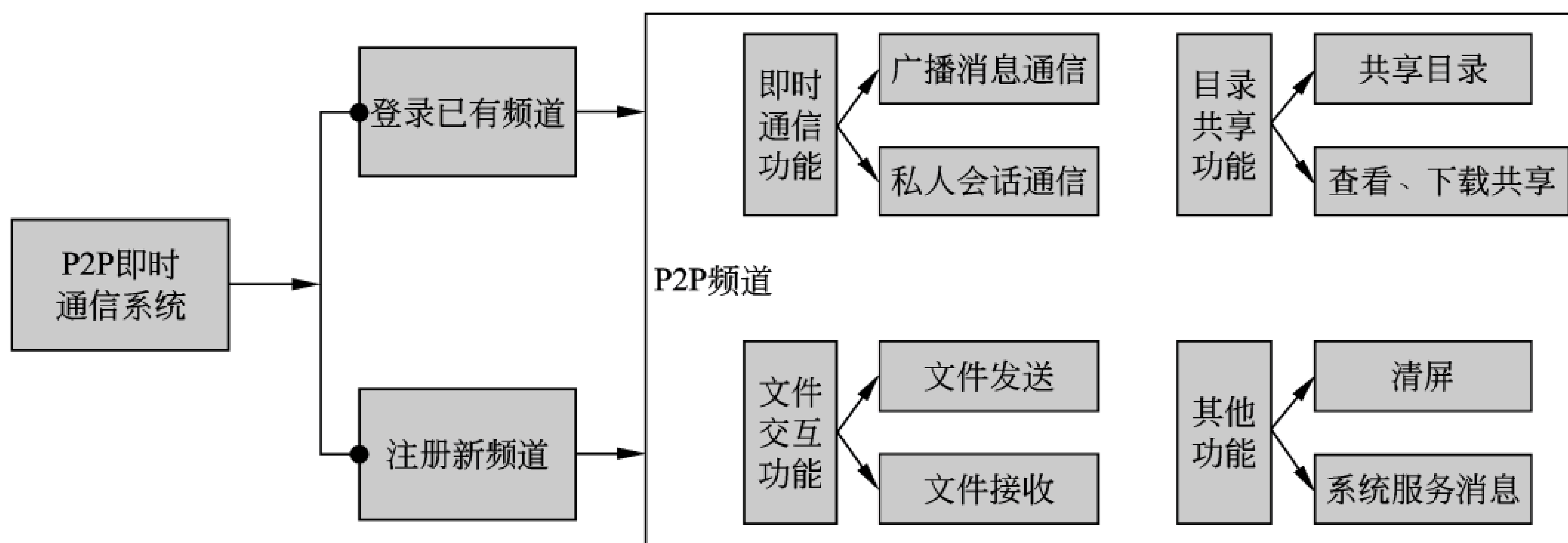


图 12.2　P2P 即时通信系统的功能结构图

12.2　系统需求分析

以上分析了系统的 4 个层次的总体结构以及整个系统的功能结构，要将这些层次与系统的功能进行统一的组织并有效地协调起来，就需要在不同的层次上实现不同的功能，这也就是整个系统的需求所在了，本节就分析一下系统的基本需求。

12.2.1　一般需求

要想实现一个基于 P2P 技术的即时通信系统，就需要将系统的设计目标、功能需求与

整个结构层次有机地结合起来，在不同的层次上实现不同的功能，这样整个系统的需求也就确定了。

1．界面显示模块

界面指的是即时通信系统在 PC 上运行时所展示的图形用户接口，程序需要一个界面来提供各种操作的入口，利用界面来直观地显示系统的运作过程，而且界面显示的结果需要简单、直观、大方，布局合理。

根据本系统要实现的功能，除了主界面外，每一个不同的功能对应着不同的操作界面，这些界面都由界面显示模块来统一完成，主要包括以下几项。

- 开始界面：程序启动后的第一个界面，供用户加入或创建一个 P2P 的通信频道。
- 主界面：整个即时通信系统的主界面，包括聊天信息的展示、功能菜单、Peer 列表等。
- 文件接收界面：当有 Peer 向你发送文件时，单击文件接收按钮就是弹出文件接收的界面，可以显示文件内容，也可以将文件存储到本地。
- 私人聊天界面：当两个 Peer 间进行私人聊天时，会弹出一个私人的聊天界面，这两个 Peer 间就可以进行私人聊天，聊天信息对其他 Peer 而言是不可见的。
- 文件共享界面：主要用于 Peer 共享文件的操作，可以显示并查看共享目录和文件的详细信息，并提供对共享文件的操作。
- 文件选择对话框：准确地说它并不是系统所有的显示界面，因为本系统中要用到文件发送和接收、文件共享、文件存储等操作，所以，文件选择对话框的功能主要是方便用户浏览选择文件路径和目录。

以上都是界面显示模块需要完成的基本功能。

2．功能模块

就整个系统而言，功能有很多，如数据交互、文件共享、私人聊天等，在这些功能中，文件共享和私人聊天在系统运行中，需要弹出单独的操作界面来执行这两个功能，这样，就需要有独立的功能子模块去进行实现。所在，在功能模块里，主要就实现两个功能，分别是文件共享的功能和私人聊天功能。

- 文件共享功能：此功能包括两个方面，一方面是主动共享本地的目录，设置共享。另一方面是查看其他 Peer 的共享目录，也就是查看共享。
- 设置共享：当用户单击主界面中的“文件共享”按钮时，弹出文件选择对话框，用户可以自由选择本地硬盘中用于共享的目录，选择完成后，此目录下的文件就全部共享出去了，其他的 Peer 就可以查看。
- 查看共享：需要查看当前 P2P 频道中存在的共享目录时也很简单，只需在 Peer 列表中，选择你要查看的 Peer 结点并右击，然后在弹出的快捷菜单中选择“查看共享目录”选项即可，这时就会弹出此 Peer 共享目录的情况。
- 频道广告功能：向网络中进行定时的广播，将频道信息广播给所有的结点。
- 私人聊天功能：私人聊天就是实现纯粹的 P2P 通信，整个通信过程在两个 Peer 之间完成，无须服务器的中转。在 Peer 列表中，选择你要与之通信的 Peer 结点名称

并右击，在弹出的快捷菜单中选择“进行私人通信”选项，就会弹出一个聊天对话框。用户就可以在此窗口中进行一对一的即时会话了。

3．消息模块

消息模块在本系统中是一个抽象的概念，系统中所有需要由网络发送的东西，都由消息模块来完成。根据系统的设计目标，在消息模块中要实现的基本需求如下：

- 频道消息：在本系统中是通过一些 P2P 的频道来模拟 P2P 网络的，每个 P2P 频道就是一个微型的 P2P 网络。Peer 结点与频道之间的交互都由频道消息来完成，因而，频道消息简单地说就是 Peer 结点发送到 P2P 频道中的文本消息，包括频道名称信息、用户输入、输出信息等。
- 私有聊天消息：私有聊天消息主要是两个 Peer 结点之间的信息交互，是两个 Peer 结点之间相互发送的文本消息
- 分享列表消息：此消息主要用于文件共享的功能上，确切地说它并不是一个真正的消息，而是一个特定的结构，用来告诉 P2P 频道中的每个 Peer 结点当前共享的文件信息。可以想象成它类似于这样一条消息：“嗨，朋友们，这是我的共享文件清单：XXX，YYY……”。
- 文件消息：此消息主要用于文件的交互传输，描述了一个文件信息被一个 Peer 结点发送到网络中的消息，当然，这个消息中也包括了文件的内容。
- 服务消息：是由系统发送的消息，用于消息的同步和管理，

4．网络模块

网络模块主要用于实现底层的 P2P 机制，通过基于 P2P 的网络实现机制，使整个即时通信系统完全架构在点对点的基础上，系统不再有中心服务器的概念，消息在两个对等的结点之间交互，再也无须服务器的中转和转发。网络模块主要通过多播机制来实现的。

- 网络分发：主要用于通过网络来发送消息，它并不真正关心消息是否被发送出去，而具体的消息发送由多点网络分发来完成。
- 多点网络分发：它是一个真正的用于发送消息的功能模块，通过多点的 Sockets 向网络中发送数据。

12.2.2　系统通信用例分析

系统通信模块，包括 3 个大的通信流程，当用户登录与注册时，需要完成 Peer 与系统之间的通信，在同一频道内，有的 Peer 之间通过广播的方式进行通信，两个 Peer 之间还有一个私人会话的通信。这 3 个通信过程的用例图及说明如下。

1．用户的登录与注册

当 Peer 结点要加入到聊天系统中时，有两种方法，一种是登录一个已有的频道；另一种是注册一个新的频道，用户的登录与注册的用例图如图 12.3 所示。

在图 12.3 中的用例图中，首先是一个名为 new_peer 的用户启动系统，登录或是注册一个 P2P 频道。如果是登录一个已有频道，那么 new_peer 就加入到此频道的 Peer 列表中；

如果是注册一个新频道，则其他的 Peer 点就可以在输入新频道的认证密码后加入到此频道中。

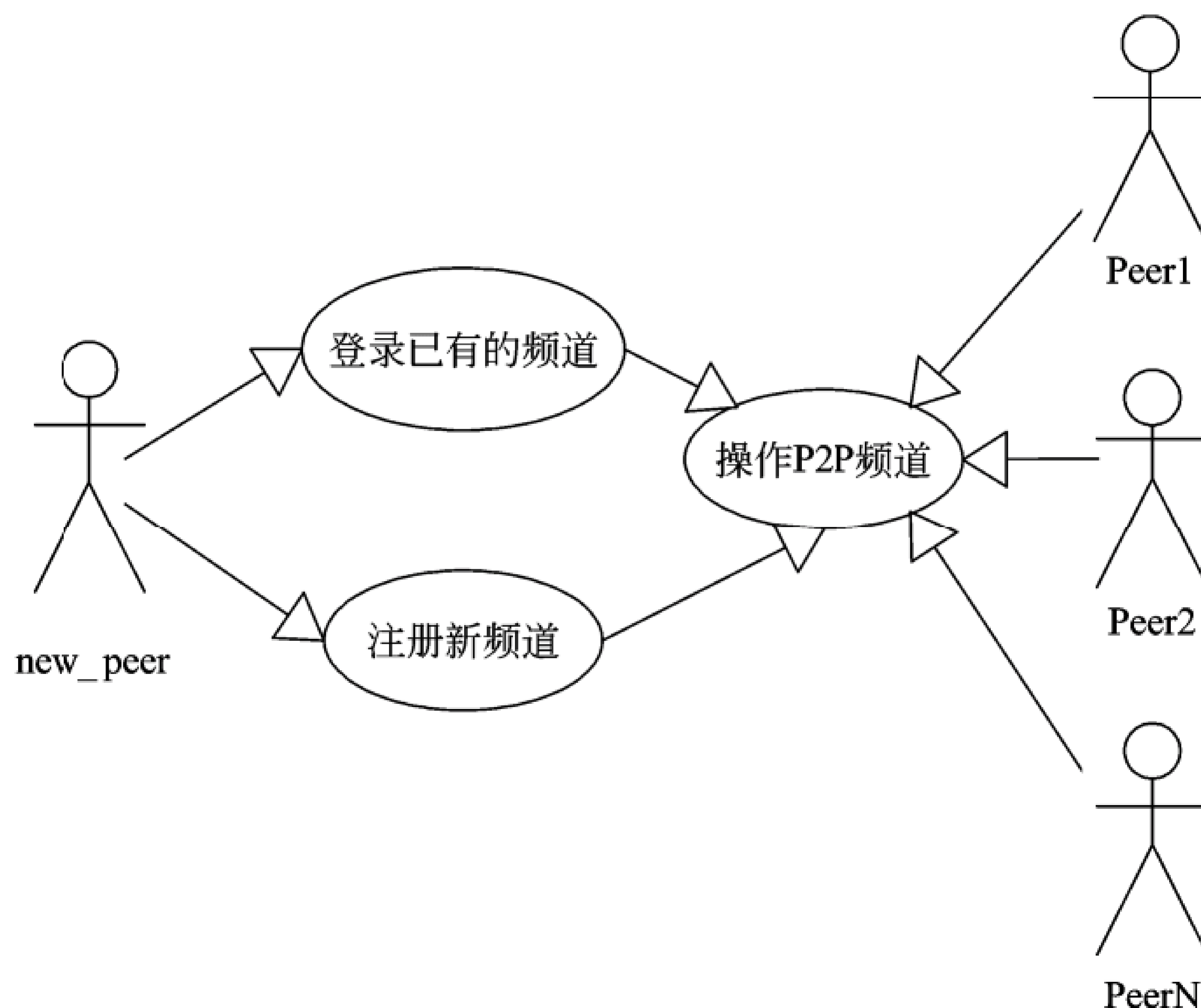


图 12.3　新 Peer 结点登录或注册频道时的用例图

2．同一频道内的广播会话

在同一频道内，所有 Peer 间的会话是以广播的形式进行的，也就是说频道内某一 Peer 发出的消息能被同一频道内其他所有 Peer 接收，频道内的广播会话用例图如图 12.4 所示。

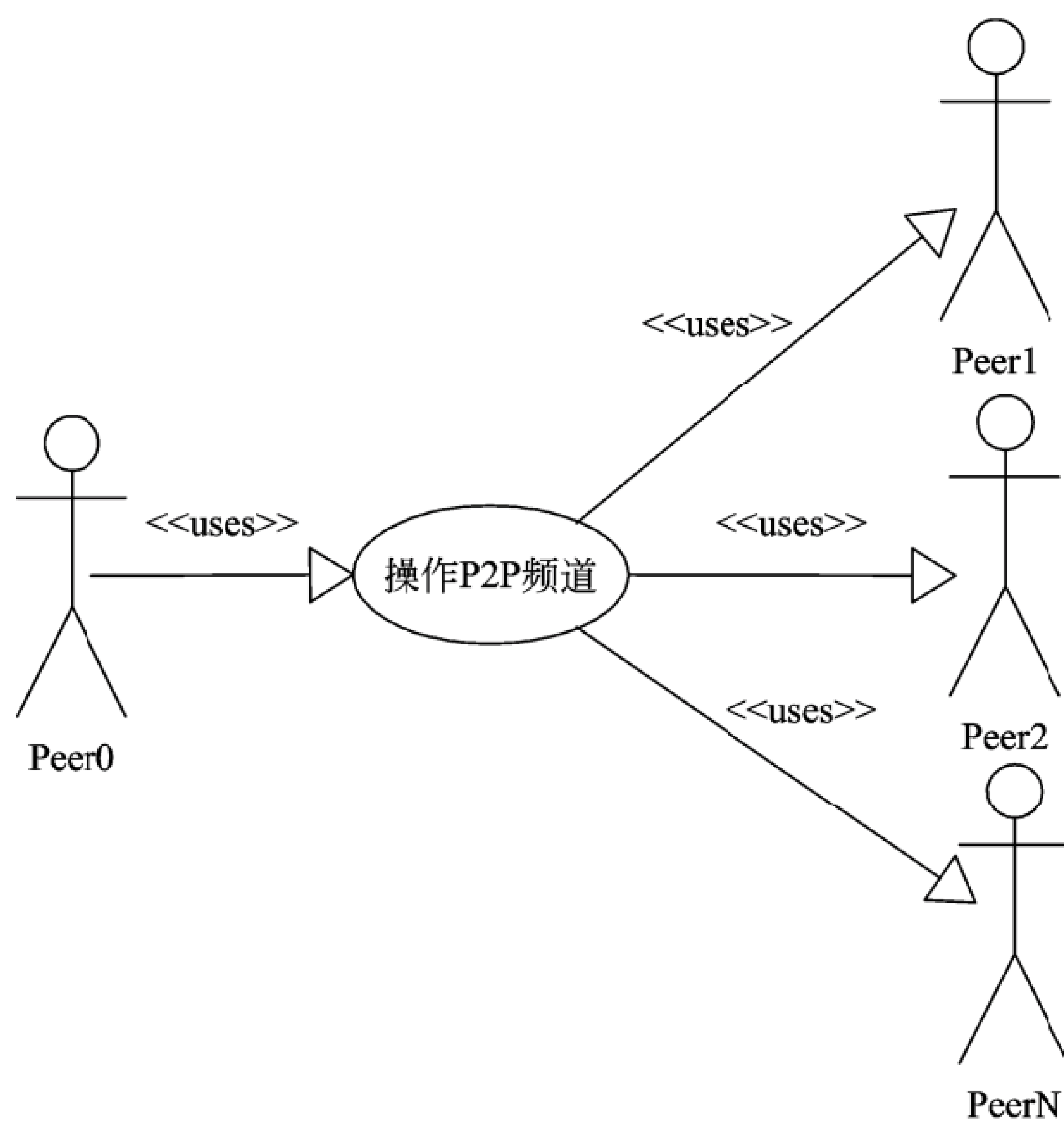


图 12.4　同一频道内的广播通信用例图

在图 12.4 所示的这个广播通信过程中，一个 Peer0 通过操作 P2P 频道，向频道内发送一条消息，频道会将这条消息广播给 Peer 列表中其他所有的 Peer，这样，Peer0 的消息就借助频道的平台，以广播的形式发给每一个 Peer 了。

3．Peer间的私有会话

Peer 间的私有会话，指的是两个 Peer 间的独立会话，由两个对等的 Peer 直接建立连接，进行会话，它们之间的会话内容对其他 Peer 而言是不可见的。Peer 间的私有会话用例图如图 12.5 所示。

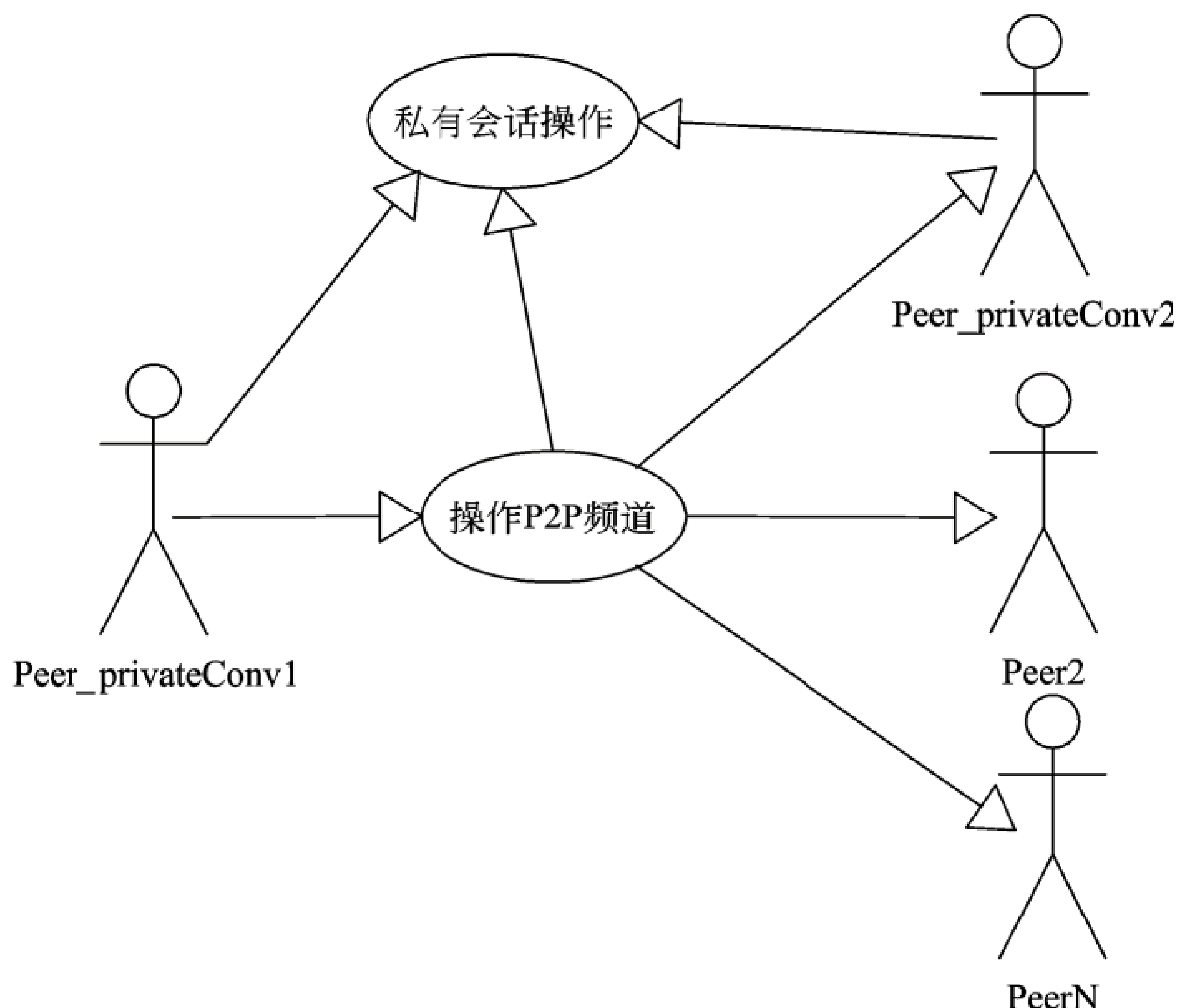


图 12.5　Peer 间的私有通信用例图

在图 12.5 所示的用例图中，Peer_privateConv1 通过操作 P2P 频道开启一个私人会话的平台，同时借助 P2P 频道与另一个 Peer_privateConv2 建立连接，这样，两个 Peer 之间就可以相互通信了。同时，他们通过私有会话操作，就可以在私人会话平台上进行独立于 P2P 频道的消息的交互了。

12.2.3　系统文件传输用例分析

在本案例的 P2P 即时通信系统中，文件传输主要表现在 3 个方面，一个是文件的发送，就是一个 Peer 将文件广播发送出去；另一个是文件的接收，Peer 结点将发送的文件接收到并执行查看、存储等操作；还有一个就是文件的共享，这个过程也涉及文件的传输，所以本节主要讲解系统文件传输的过程及关系。

1．文件发送

本系统中的文件发送是以广播形式发送的，也就是某一 Peer 发送的文件，可以同时被其他的所有 Peer 接收到，文件发送的用例图如图 12.6 所示。

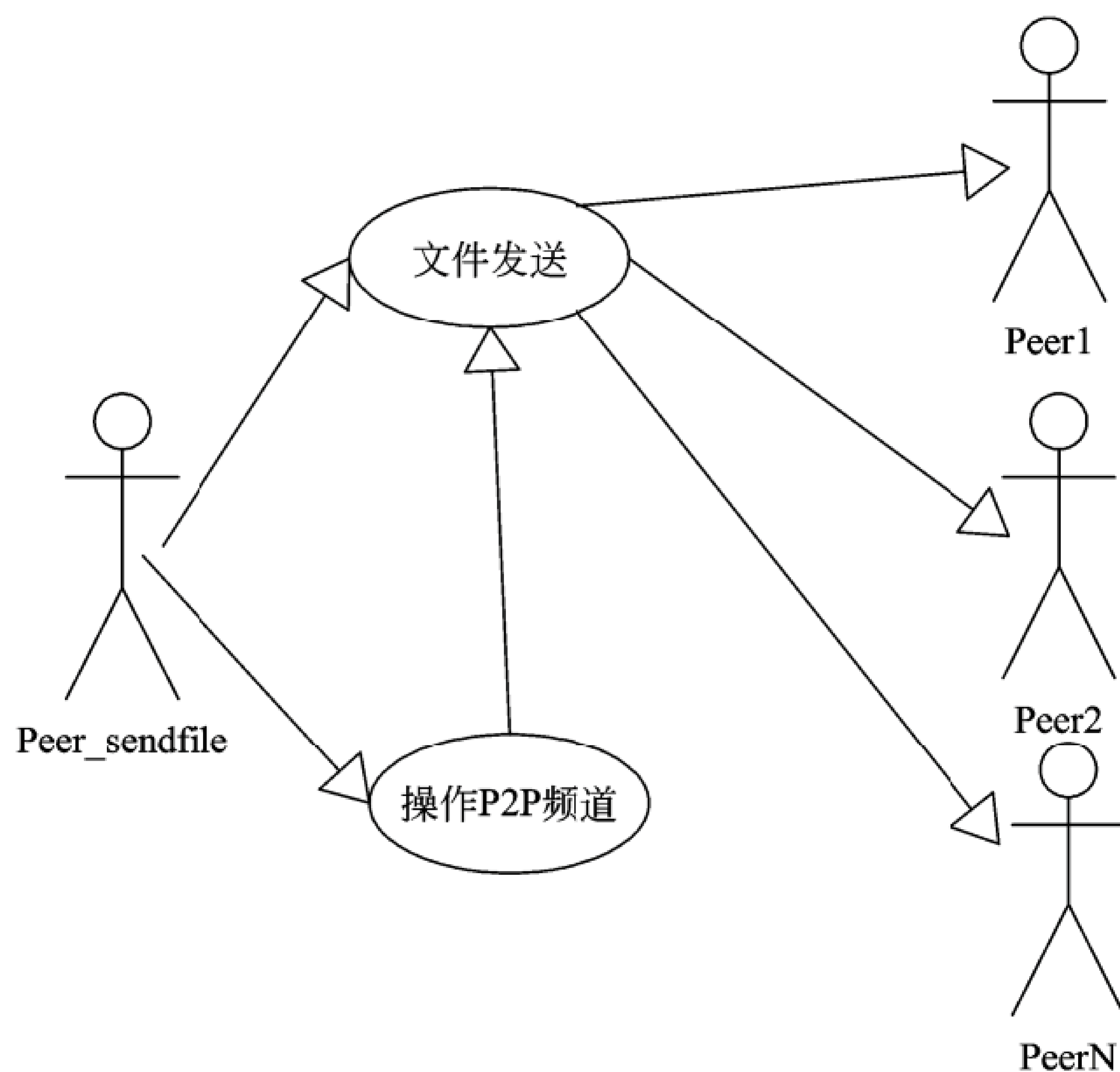


图 12.6　文件发送用例图

在图 12.6 所示的用例图中，一个 Peer_sendfile 结点，通过操作 P2P 频道执行文件发送的功能，一旦文件发送出去以后，此文件的内容就会广播给所有的其他 Peer。这样，从 Peer1 到 PeerN 中的每个 Peer 包括发送者自身，都能接收到由 Peer_sendfile 结点发送的文件了。

2. 文件接收

当一个 P2P 频道中有 Peer 发送文件时，其他的任何 Peer 都可以执行文件接收的功能了，接收文件的用例图如图 12.7 所示。

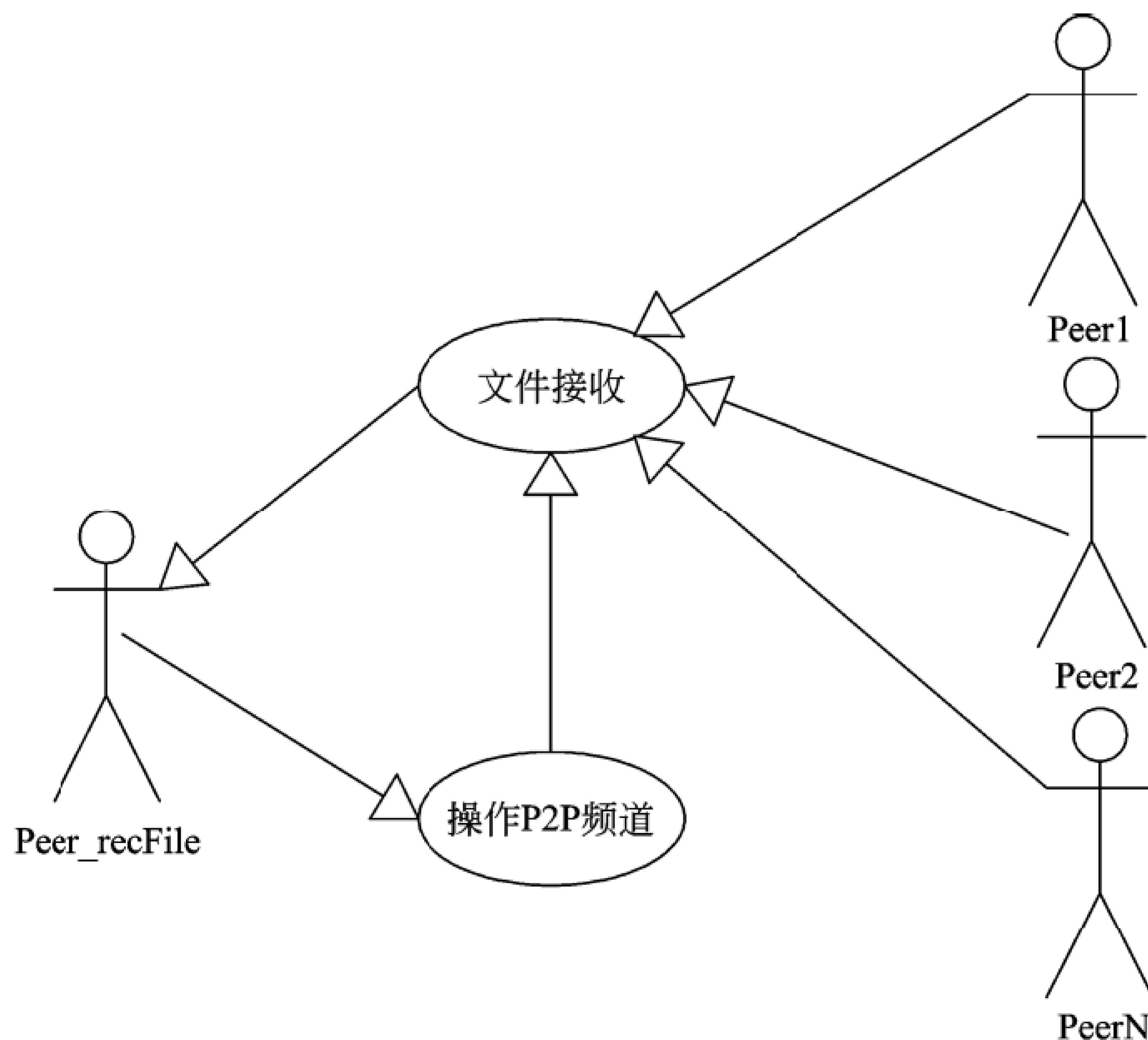


图 12.7　文件接收的用例图

在图 12.7 所示的用例图中，Peer_recFile 结点，通过操作 P2P 频道执行文件接收的功能，它所接收的文件是来自 Peer 列表中所有 Peer 发送的文件集合。

3．文件共享

本系统文件的共享有两个过程，一个是将本地目录共享出去；另一个是查看其他 Peer 的共享内容。文件共享的用例图如图 12.8 所示。

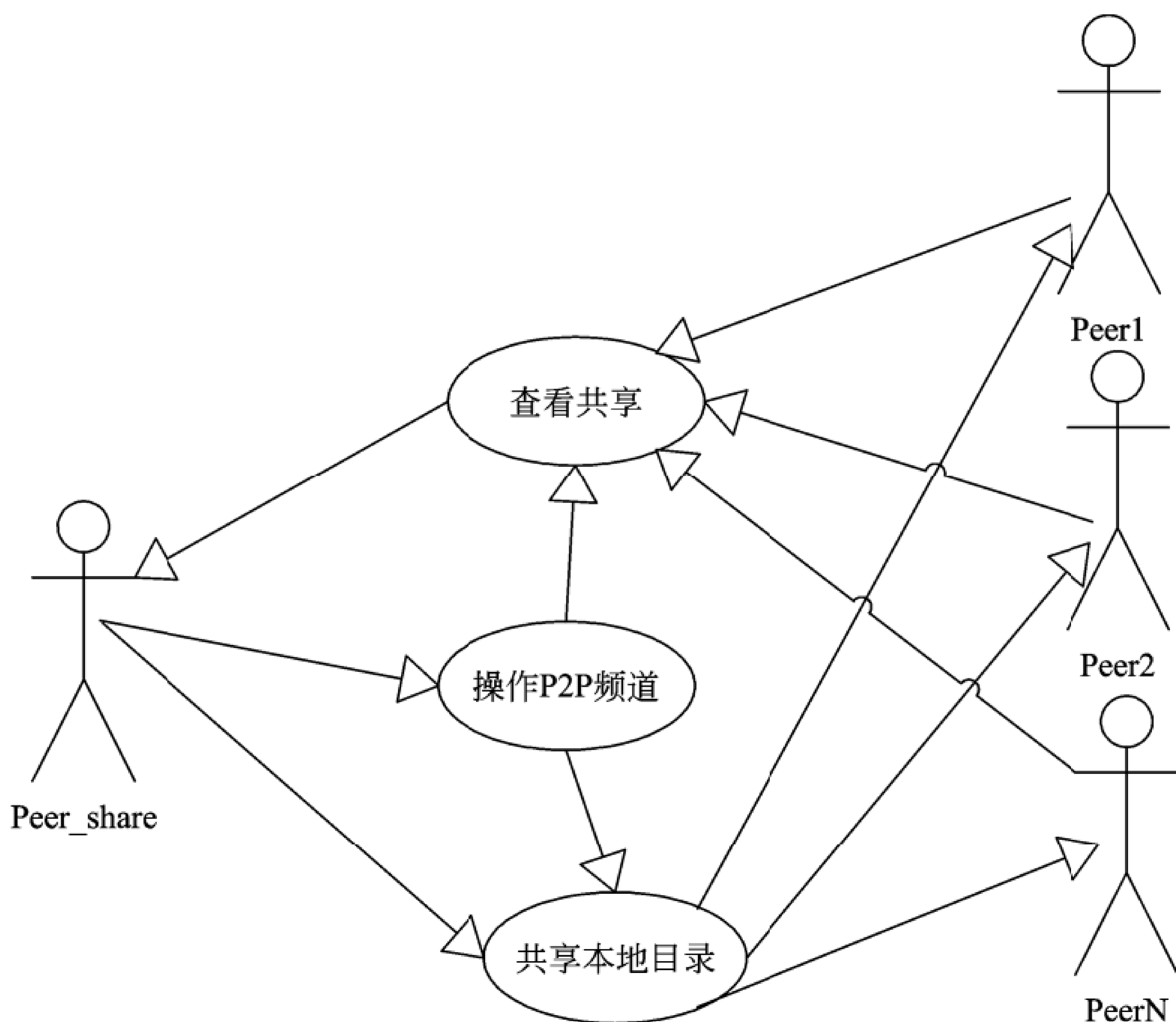


图 12.8　文件共享用例图

在图 12.8 所示的用例图中，当 Peer_share 要共享本地目录时，通过操作 P2P 频道执行共享本地目录的功能，将本地目录共享出去。当执行此共享操作以后，其他的所有 Peer 都可以浏览 Peer_share 的共享目录，并可进行查看共享文件内容、将共享文件存储到本地等操作。

当 Peer_share 想要查看共享的时候，可以通过操作 P2P 频道执行查看共享的功能，这样就可以查看 Peer 列表中每一个 Peer 所共享的目录及文件内容等信息。查看共享文件与主动共享本地目录，是一对互操作的过程。

12.3　系统的关键技术及实现机制分析

要开发一个基于 P2P 的即时通信系统，首先要解决的就是如何实现底层的 P2P 机制，其次要解决端到端的通信问题、基于网络的数据交互问题等。在文件的传输和存储上还要处理文件的读写问题，系统运行的时候要以界面的形式展现出来，所以还要用到界面编程

的知识等。总之，要顺利地开发出本系统，以上所说的一些技术问题是必须要解决的。本章就讲解一下实现本系统的关键技术，分析其实现的机制。

12.3.1　系统开发语言及实现环境

在具体讲解这些关键技术之前，首先要确定本系统的开发语言、开发环境等。本书所有的案例都是基于 Java 语言开发实现的，所以，本系统也是用 Java 开发而成，下文所说的关键技术自然都是相对于 Java 编程而言的。

本系统的实现环境要求很低，基本上能进行 Java 项目开发的编程环境都可以实现本系统，本文中系统开发实现的环境具体如下。

- 系统平台：Windows XP sp2；
- Java 平台：JDK 1.6；
- 开发工具：Eclipse；
- 第三方软件：fatJar 插件、Jar2Exe 工具等；
- 硬件要求：对硬件无特殊要求，只要可以运行 Eclipse 和 Jdk 1.6 的主机，都可以开发本系统；
- 网络要求：由于本系统的测试和验证需要在网络条件下进行，所以至少有两台以上的主机互联的网络环境。

注意： fatJar 插件是一个 Jar 文件的打包工具，Jar2Exe 是一个将 Java 文件生成 exe 文件的工具。这两个软件是为程序的打包和发布而用的，只针对需要将程序发布出去的用户的需要，与系统的开发过程无关，没有这两个打包工具不会对系统开发有任何影响。

12.3.2　系统中的 P2P 实现机制

P2P（Peer-to-Peer　端到端）模型是与 C/S（客户/服务器）模型相对应。基于 C/S 的用户间通信需要由服务器中转，而基于 P2P 的用户间通信则是直接通信，去掉了服务器这一层，因而本系统最显著的特点是，没有中心服务器。

1．结点发现的基本方法

在 P2P 网络中，任意两个端点之间可实现直接通信。在基于 C/S 的网络中，客户端可以通过向服务器注册来实现彼此之间的定位（获得 IP 和端口）。也就是说，网络中的任意两点，只有获得其 IP 地址和端口号，那么这两点之间就可以相互通信了。

假设有一个端点 A，欲和 P2P 网络中的其他端点通信，在通信之前，端点 A 必须首先把自己的 IP 和端口通知 P2P 网络中的其他每一个端点。其他每个端点收到这个信息后，就获得了端点 A 的 IP 和端口，随后向端点 A 反馈自己的 IP 和端口信息，使端点 A 也获得 P2P 网络中每个端点的 IP 和端口。

以上是个很简单的通信过程，在这个过程中要实现对等点发现服务有多种方法。最简

单的机制是显式的点到点配置。这种机制通过要求每个对等点知道所有它可能与之交互的其他对等点，并与它们相连，来进行工作。

还有一种发现结点的方法是使用中央目录作为中介。该方法在许多传统的、非 P2P 分布式类型的应用程序中间很流行，在这种方法中，对等点向中央目录注册自己的存在，并使用中央目录定位其他对等点。

许多流行的 P2P 应用程序使用的网络模型并不是中央目录结构的，在网络模型中，单个对等点只知道局域网络上的对等点身份。每个对等点都作为那些与之相连的对等点的目录。对等点通过向相邻对等点传播目录查询并返回相关的响应来进行合作。

上面 3 种机制有无数种变体。不讨论这些变体了，让我们继续前进并研究另一种发现机制。这种机制就是 IP 多播发现。

2．IP多播发现

就每个对等点维护自己的目录这点而言，多播模型类似于网络模型。但是，对等点不通过合作来实现大规模网络查询。另外，对等点利用网络本身提供的特性（IP 多播）来定位和标识其他对等点。

IP 多播是无连接和不可靠的（不像 TCP/IP 是面向连接和可靠的）。虽然它使用 IP 数据报；但是不像单播 IP 数据报那样是从一台主机发送到另一台主机，多播 IP 数据报可以同时发往多台主机。

对等点定期使用 IP 多播来宣布自己的存在，在宣布的信息中包含了它们的主机名和一个用于正常通信的端口。对此消息感兴趣的对等点检测这个消息后，抽取出主机名和端口号，并使用该消息建立一个通信通道。

由以上的特点可以知道，多播技术是一种允许一个或多个发送者（多播源）发送单一的数据包到多个接收者（一次的，同时的）的网络技术。多播源把数据包发送到特定多播组，而只有属于该多播组的地址才能接收到数据包。多播组中的端点（主机）可以是在同一个物理网络，也可以来自不同的物理网络（如果有多播路由器的支持）。因此，多播技术是本系统对 P2P 实现机制的选择。

3．在Java中发送和接收多播信息的方法

要想用 Java 编程实现发送多播信息，需要经历几个基本的编程步骤。能完成以下这几步，就是可以用 Java 程序来实现多播信息的发送了。

（1）确定发送的具体信息内容。

```
String msg = "Hello";
```

（2）选用专门为多播指定的 D 类 IP 地址（224.0.0.1 到 239.255.255.255），创建一个多播组。

```
InetAddress group = InetAddress.getByName ("228.5.6.7");
```

（3）使用指定的端口（一般选 1024 以上的端口号）建立多播套接字。

```
MulticastSocket s = new MulticastSocket (6789);
```

（4）加入多播组。

```
s.joinGroup (group);
```

（5）创建一个数据报封装多播信息。

```
DatagramPacket hi = new DatagramPacket(msg.getBytes(), msg.length(), group,
6789);
```

（6）消息发送。

```
s.send (hi);
```

对多播发送出去的消息，还必须要进行接收，只有发送和接收都完成了，多播消息的交互才算完成了，以下是接收多播信息的步骤。

（7）开辟接收缓冲区。

```
byte[] buf = new byte[1000];
```

（8）创建接收数据报。

```
DatagramPacket recv = new DatagramPacket(buf, buf.length);
```

（9）接收消息。

```
s.receive (recv);
```

注意：以上发送和接收程序在同一个文件中实现，若在不同文件中实现则应分别定义多播套接字并加入多播组。

以上就是 Java 中发送和接收多播信息的方法。

4．简单的多播发送和接收模型的实现

下面用一个简单的示例演示两个进程如何使用 IP 多播进行通信，这两个进程一个是发送端进程，主要是发送多播消息；另一个是接收端进程，用来接收发送端发出的多消息。本文后面所讲的 P2P 实现机制，都是基于这两个进程来实现的。

在本例中，接收端进程进行循环并等待数据报包的到来。每接收到一个包，接收端就会向控制台打印一条简短的诊断消息。而发送端的角色要简单得多，它在多播完单个数据报包后就直接退出。

以下两个示例在本章工程目录的 test 文件夹里，随书光盘的源代码位置为：

【示例源代码：\源代码\ch12\ch12_code\p2pchat\src\p2p\chat\test\Sender.java】

Sender 类，主要用来实现发送端进程，演示如何通过多播方式将一条消息发送出去，部分程序代码如下：

```
 /**
  *Send 类，IP 多播在发送端进行消息发送的方法，主要用来演示一下，在网络中如何将一条消息
  *以 IP 多播的方式发送到网络中。
  */
package p2p.chat.test;                         //声明包的路径
import java.net.*;                             //引入程序所需的开发包
//定义一个 IP 多播发送端的程序，用来通过 IP 多播机制发送一条消息
public class Sender {
   public static void main(String[] args) {
```

```
        try {
            //发送一个“hello”的字符串消息，需要将此消息转换成字节形式
            byte[] arb = new byte[] { 'h', 'e', 'l', 'l', 'o' };
            //通过 InetAddress 的 getByName 方法，在给定主机名的情况下确定主机的 IP
            //地址，此 IP 地址设置为多播所用 IP: 230.0.0.1
            InetAddress inetAddress = InetAddress.getByName ("230.0.0.1");
            //通过 DatagramPacket 类，来封装一个包含要发送消息的数据包，需要传入要
            //发送消息的内容，消息的长度、多播 IP、端口等参数，本程序端口设定为：7777
            DatagramPacket datagramPacket = new DatagramPacket(arb,arb.
            length,inetAddress, 7777);
            //通过 MulticastSocket 类，新建一个多播套接字对象
            MulticastSocket multicastSocket = new MulticastSocket ();
            //直接调用 multicastSocket 的 send 方法，将封装的数据报发送出去
            multicastSocket.send (datagramPacket);
        //捕获并处理异常信息
        } catch (Exception exception) {
            exception.printStackTrace ();
        }
    }
}
```

以上所演示的就是在 IP 多播网络中在发送端发送一个多播消息的示例，这条消息发送出去以后，就需要对应的接收端来接收这条消息。下面讲一下如何在接收端来接收多播消息。

与发送端 Sender 类对应的是接收端 Receiver 类，源代码位置：

【示例源代码：\源代码\ch12\ch12_code\p2pchat\src\p2p\chat\test\Receiver.java】

Receiver 类，主要用来实现接收端的进程，通过加入多播组的方式来接收来自发送端的消息，程序代码如下：

```
/**
  *Receiver 类，IP 多播在接收端接收消息的方法，主要用来演示在网络中如何接收一条多播消
  息的过程，具体实现方法如下：
  */
package p2p.chat.test;                          //声明包的路径
import java.net.*;                              //引入程序所需的开发包
//定义一个 Receiver 类，用来在 IP 多播网络中接收一条多播消息
public class Receiver {
    /**要在 IP 多播网络接收一条消息，其主要实现方法是，通过加入到 IP 为 230.0.0.1，端
    口为 7777 的多播组中，并通过创建一个多播套接字来接收 IP 多播数据报***/
    public static void main(String[] arstring) {
        try {
            //创建多播套接字并将其绑定到特定端口，这里要与发送端口一致，设定为：7777
              MulticastSocket multicastSocket = new MulticastSocket (7777);
            //通过 InetAddress 的 getByName 方法，在给定主机名的情况下确定主机的
              IP 地址，此 IP 地址设置为多播所用 IP：230.0.0.1
            InetAddress inetAddress = InetAddress.getByName ("230.0.0.1");
            //通过多播套接字的 joinGroup 方法，依据指的 IP 加入多播组
            multicastSocket.joinGroup (inetAddress);
            //通过一个无限循环来接收来自发送端的消息
            while (true) {
                //开辟一个字节数组缓冲区，用于存放接收结果，这里设置为 100 字节大小
                byte[] arb = new byte[100];
                //根据缓冲区名和长度，来新建一个数据报
```

```
                DatagramPacket datagramPacket = new DatagramPacket(arb,
                arb.length);
                //调用多播套接字的 receive()方法来接收数据报
                multicastSocket.receive (datagramPacket);
                //将接收到的消息转换成字符串打印输出
                System.out.println (new String(arb));
            }
        //捕获并处理异常信息
        } catch (Exception exception) {
            exception.printStackTrace ();
        }
    }
}
```

注意：在程序中多处用到了 MulticastSocket 类，这个类叫做多播数据报套接字类，主要用于发送和接收 IP 多播包。MulticastSocket 是一种（UDP）DatagramSocket，它具有加入 Internet 上其他多播主机的“组”的附加功能。多播组通过 D 类 IP 地址和标准 UDP 端口号指定。D 类 IP 地址在 224.0.0.0 和 239.255.255.255 的范围内（包括两者）。地址 224.0.0.0 被保留，不应使用。在实际使用中，可以通过首先使用所需端口创建 MulticastSocket，然后调用 joinGroup（InetAddress groupAddr）方法来加入多播组。

以上的 Sender.java 类与 Receiver.java 类，就是在 IP 多播网络中发送端发送消息和接收端接收消息的实现过程，本系统在实现 P2P 机制的时候，就是按以上的原理来实现的。

注意：Sender.java 与 Receiver.java 两个程序的源码位于 p2pchat 工程目录下 p2p.chat.test 包下。关于 p2pchat 工程就是本系统的开发工程名，在后文会讲到。

12.3.3　Java 网络编程技术

作为一个即时通信系统，需要完成很多网络底层的数据发送与接收，如数据报的发送、消息的交互、控制消息、数据消息的传输等，这些都需要用网络编程技术来完成，因而本节就讲一下与本系统有关的 Java 网络编程技术。

根据上文对 P2P 实现的分析，在网络编程层面上主要用到 MulticastSocket 的多点广播方法和 DatagramPacket 接收数据的方法。

1．使用MulticastSocket实现多点广播

在 java.net 包中，MulticastSocket 类可以将数据报以广播方式发送到数量不等的多个客户端。

若要使用多点广播时，则需要让一个数据报标有一组目标主机地址，当数据报发出后，整个组的所有主机都能收到该数据报。IP 多点广播（或多点发送）实现了将单一信息发送到多个接收者的广播，其思想是设置一组特殊网络地址作为多点广播地址，每一个多点广播地址都被看作一个组，当客户端需要发送、接收广播信息时，加入到该组即可。

上文已经讲过，IP 协议为多点广播提供了这批特殊的 IP 地址，这些 IP 地址的范围是

224.0.0.0～239.255.255.255。多点广播的示意图如图 12.9 所示。

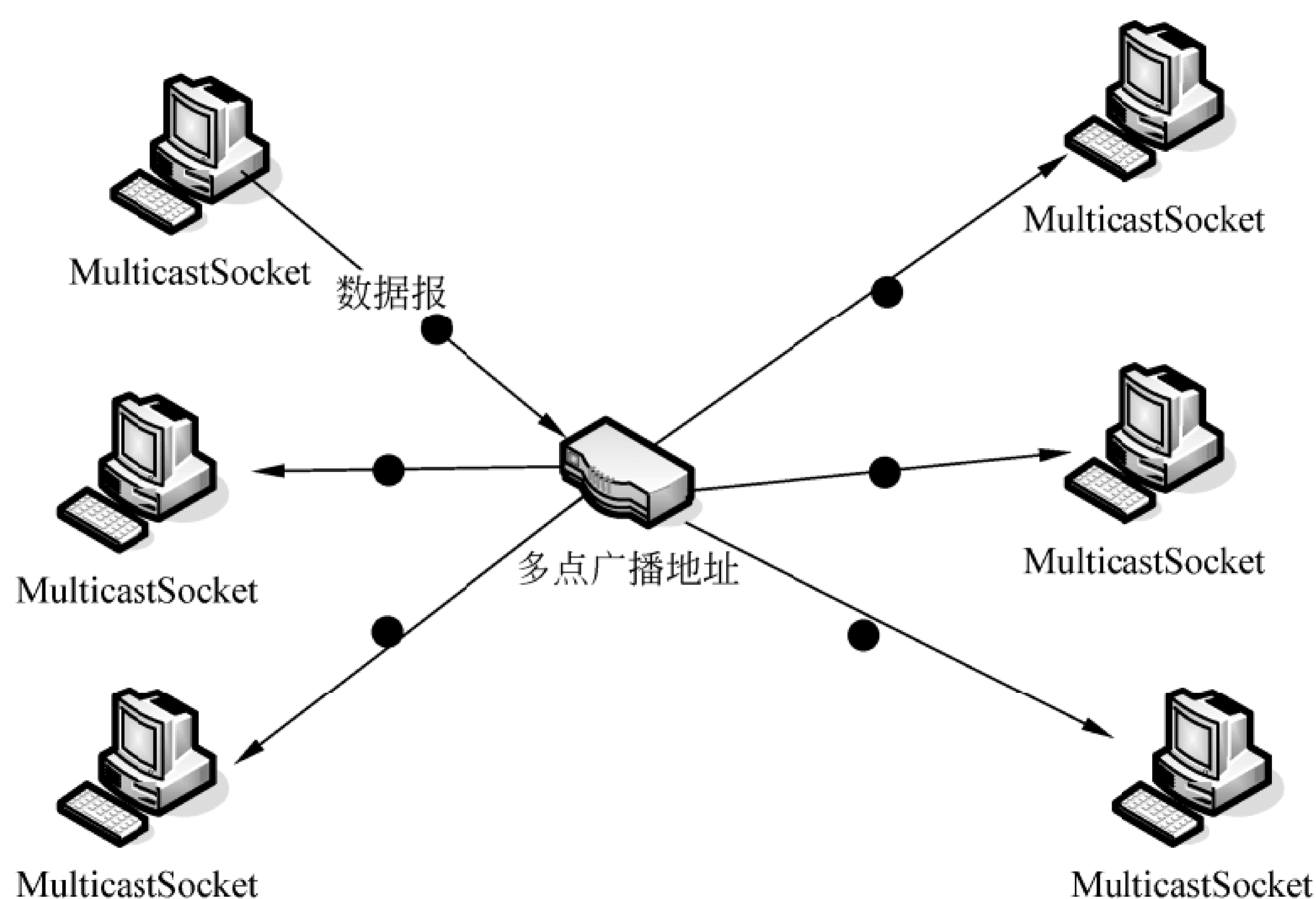


图 12.9　多点广播的示意图

从图 12.9 中可以看出，通过 Java 实现多点广播时，MulticastSocket 类是实现这一功能的关键，当 MulticastSocket 把一个 DatagramPacket 发送到多点广播 IP 地址时，该数据报将被自动广播到加入该地址的所有 MulticastSocket。MulticastSocket 类既可以将数据报发送到多点广播地址，也可以接收其他主机的广播信息。

要发送一个数据报时，可使用随机端口创建 MulticastSocket，也可以在指定端口创建 MulticastSocket。MulticastSocket 提供了如下 3 个构造器：

- 使用本机默认地址、随机端口来创建一个 MulticastSocket 对象。

```
public MulticastSocket()
```

- 使用本机默认地址、指定端口来创建一个 MulticastSocket 对象。

```
public MulticastSocket(int portNumber)
```

- 使用本机指定 IP 地址、指定端口来创建一个 MulticastSocket 对象。

```
public MulticastSocket(SocketAddress bindaddr)
```

创建一个 MulticastSocket 对象后，还需要将该 MulticastSocket 加入到指定的多点广播地址，MulticastSocket 使用 jionGroup()方法来加入指定组；使用 leaveGroup()方法脱离一个组。方法如下：

- 将该 MulticastSocket 加入指定的多点广播地址。

```
joinGroup (InetAddress multicastAddr)
```

- 让该 MulticastSocket 离开指定的多点广播地址。

```
leaveGroup (InetAddress multicastAddr)
```

如果创建仅用于发送数据报的 MulticastSocket 对象，则使用默认地址、随机端口即可。但如果创建接收用的 MulticastSocket 对象，则 MulticastSocket 对象必须具有指定端口，否

则发送方无法确定发送数据报的目标端口。

使用 MulticastSocket 进行多点广播时所有通信实体都是平等的，也就是说通信实体之间是 Peer-to-Peer 的，它们都将自己的数据报发送到多点广播 IP 地址，并使用 MulticastSocket 接收其他人发送的广播数据报。

2．用DatagramPacket来发送和接收数据

DatagramPacket 对象就是数据报的载体。如果用来接收数据，用下面这个方法创建：

```
public DatagramPacket(byte []buf, int length)
```

在这个方法中，buf 是存放数据的字节型数组，length 是能接收的最大长度。

如果是发送数据报，用下面这个方法：

```
public DatagramPacket(byte []buf, int length, InetAddress address, int port)
```

在这个方法中，后两个参数分别是目的地址和端口。

以上就是 DatagramPacket 发送和接收数据的方法。此类还有一些常用的方法，这里一并列出，供读者参考。

- public byte[] getData()：获取存放在数据报中的数据。
- public int getLength()：获取数据的长度。
- public InetAddress getAddress()：获取数据报中的 IP 地址。
- public int getPort()：获取数据报中的端口号。
- public void setData(byte []buf)：设置数据报中的内容为 buf 所存储的内容。

DatagramPacket 的用法比较简单，本系统中也是用到了它的一些很简单的功能，所以读者只需了解即可。

12.3.4　Java 的加密编程技术

本系统中某些消息的分发以及用户登录的验证都需要用密文进行传输，这就涉及 Java 的加密编程技术。下面就简要地讲解一下 Java 中的加密编程。

1．Java密码体系

Java 密码（Cryptography）体系依赖于 JCA 和 JCE。Java Cryptography Architecture（JCA）和 Java Cryptography Extension（JCE）是两个非常重要的框架，它们提供了非常简洁通用的 API 接口，接口跟实现是完全分离的。JCA 包括了数字签名和消息摘要的 API，JCE 扩展了 JCA，提供了更多的安全 API。Java 针对下面一些常用的算法提供了接口和实现：

- 对称的分组加密算法，如 DES、RC2 和 IDEA。
- 对称的流加密算法，如 RC4。
- 非对称流加密算法，如 RSA。
- 基于密码的加密（PBE）。
- 密钥交换协议，如 Diffie-Hellman。
- 信息认证码（MAC）。

2．关于Java加密中的Cipher

加密是一个将欲加密的资料用一些数学运算转成一团令人看不懂的东西的过程；解密则是将加密文转换回原始文字的过程。这个过程中，扮演原始文字与加密文字之间转换的数学算法称为 Cipher。

Cipher 一般会用 Key 来加密与解密资料。所谓 Key 是指一个机密值，我们可将它视为一个通行密码。加密文字必须使用对应的 Key 才能解密为原始文字。Cipher 有几种类型，它几种类型是：

- ❑ 对称型 Cipher：对称型 Cipher 在传送端与接收端所用的 Key 是一样的，对称型 Cipher 又叫 Private Key Cipher，因为 Key 的值只有传送端和接收端知道。如果有第三者知道了 Private Key 值，也就能解开加密的资料。
- ❑ 非对称型 Cipher：非对称型的 Cipher 又叫 Public Key Cipher，Cipher 除了 Private Key 外，还会引进一个可以随意散发的 Public Key。被 Public Key 加密的资料只有相对应的 Private Key 可以解开，同样地，被 Private Key 加密的资料也只有相对应的 Public Key 可以解开。
- ❑ 信息摘要（Message Digest）：信息摘要是从一组输入资料计算所得的一个特别数字，其原理运作就如 hash function 一般。在密码的运用里，一般是用来验证资料是否被篡改。

3．Java中的加密算法

Java 中的加密算法，基本分两大类，一类为单向加密算法，另一类是对称式加密算法，如基本的单向加密算法如下。

- ❑ BASE64：严格地说，其属于编码格式，而非加密算法。
- ❑ MD5：Message Digest algorithm 5，信息摘要算法。
- ❑ SHA：Secure Hash Algorithm，安全散列算法。
- ❑ HMAC：Hash Message Authentication Code，散列消息鉴别码。

另一类就是复杂的对称加密（DES、PBE）、非对称加密算法，如下所示。

- ❑ DES：Data Encryption Standard，数据加密算法。
- ❑ PBE：Password-based encryption，基于密码验证。
- ❑ RSA：算法的名字以发明者的名字命名，Ron Rivest、AdiShamir 和 Leonard Adleman。
- ❑ DH：Diffie-Hellman 算法，密钥一致协议。
- ❑ DSA：Digital Signature Algorithm，数字签名。
- ❑ ECC：Elliptic Curves Cryptography，椭圆曲线密码编码学。

以上列出了 Java 中的多种加密算法，本系统中主要使用对称的加密算法来对数据进行加密，这也很容易理解。比如，PeerA 用一个密钥对一个文件加密，而 PeerB 读取这个文件的话，则需要和 A 一样的密钥，双方共享一个私钥。这一过程可用 PEB 的加密算法来实现。

PBE——Password-based encryption（基于密码的验证）。其特点在于口令由用户自己掌管，不借助任何物理媒体。这与本系统的需求是一样的，因为本系统中不借助于任何数

据库，用户创建的频道密码由用户自己掌管，这也正符合了 PEB 的加密特点。

4．PEB加密的程序实现

PEB 采用随机数杂凑多重加密等方法保证数据的安全性，是一种简便的加密方式。它的加密、解密过程主要由 3 个步骤组成，分别为转换密钥、加密过程和解密过程这 3 步。这 3 个步骤的实现可用如下代码表示。

1）转换密钥

```
private static Key toKey(String password) throws Exception {
    //通过带有密码的构造方法来新建一个 PBEKeySpec 对象，需要字符数组型的密码参数
    PBEKeySpec keySpec = new PBEKeySpec (password.toCharArray ());
    //通过 getInstance()方法，返回转换指定算法的秘密密钥的 SecretKeyFactory 对象
    SecretKeyFactory keyFactory = SecretKeyFactory.getInstance (ALGORITHM);
    //根据提供的密钥规范（密钥材料）生成 SecretKey 对象
    SecretKey secretKey = keyFactory.generateSecret (keySpec);
    //将 secretKey 对象返回
    return secretKey;
}
```

在转换密钥过程中，用到了这样一个方法：

```
SecretKeyFactory keyFactory = SecretKeyFactory.getInstance (ALGORITHM);
```

这里的 ALGORITHM 指代的是加密算法，本系统用 PBEWITHMD5andDES 算法进行数据的加密。

SecretKeyFactory 类表示秘密密钥工厂的意思，密钥工厂用来将密钥（类型 Key 的不透明加密密钥）转换为密钥规范（底层密钥材料的透明表示形式），反之亦然。秘密密钥工厂只对秘密（对称）密钥进行操作。

密钥工厂为双工模式，即其允许根据给定密钥规范（密钥材料）构建不透明密钥对象，或以适当格式获取密钥对象的底层密钥材料。

2）加密过程

```
public static byte[] encrypt(byte[] data, String password, byte[] salt)
throws Exception {
      //调用密钥转换方法得到一个秘密（对称）密钥对象
      Key key = toKey (password);
      //为 PKCS  #5 标准中所定义的基于密码的加密法构造一个参数集合
      PBEParameterSpec paramSpec = new PBEParameterSpec (salt, 100);
      //调用 Cipher 的 getInstance()方法并将所请求转换的名称传递给它，返回一个
        Cipher 对象
      Cipher cipher = Cipher.getInstance (ALGORITHM);
      //通过传入密钥和一组算法作为参数，以初始化此 Cipher 对象
      cipher.init (Cipher.ENCRYPT_MODE, key, paramSpec);
      //按单部分操作加密或解密数据，或者结束单部分或多部分操作，并返回一个字节数组对象
      return cipher.doFinal(data);
   }
```

注意： PKCS，是一套公钥加密标准，是介于供应商之间的协议标准，可在因特网中使用公钥基础设施来提供安全的信息交流方式。而 PKCS #5，则是基于口令的密码系统规范，读者可参考密码学的相关知识。

3）解密过程

```
//解密过程是加密的逆过程，读者参考加密的过程就可以理解
public static byte[] decrypt(byte[] data, String password, byte[] salt)
throws Exception {
        Key key = toKey (password);
        PBEParameterSpec paramSpec = new PBEParameterSpec (salt, 100);
        Cipher cipher = Cipher.getInstance (ALGORITHM);
        cipher.init (Cipher.DECRYPT MODE, key, paramSpec);
        return cipher.doFinal (data);
    }
}
```

在加密解密过程中都用到了 cipher.init(Cipher.DECRYPT_MODE, key, paramSpec)方法，Cipher 类为加密和解密提供密码功能。它构成了 Java Cryptographic Extension（JCE）框架的核心。

为创建 Cipher 对象，应用程序调用 Cipher 的 getInstance()方法并将所请求转换的名称传递给它。还可以指定提供者的名称（可选）。转换，是一个字符串，它描述为产生某种输出而在给定的输入上执行的操作（或一组操作）。转换始终包括加密算法的名称（例如，DES），后面可能跟有一个反馈模式和填充方案。

本系统中，NetworkDispatcher 类的实现用到很多与 Java 加密编程技术相关的知识，理解了以上的知识对整个系统代码和实现原理的理解有重要作用。

注意：关于 NetworkDispatcher 类的知识后文会讲到。

12.3.5　Java I/O 流技术

在本系统中，要实现 Peer 间文件的交互传输、消息的输入输出，都要用到 Java IO 的相关技术，这也是理解系统实现机制的一个重要知识。本节就讲一下 Java I/O 流技术的简单应用。

Java 的 I/O 系统是用来进行输入和输出的，Java 系统本身提供了非常丰富的类库，利用这些丰富的 I/O 类库，几乎可以进行一切的 I/O 操作。

1．Java I/O的分类

- 按照流的方向划分，可以分为输入流和输出流。
- 按照流所处理的数据类型划分，可分为字节流和字符流。
- 按照流是否可以直接访问资源划分，可以分为结点流和处理流。

所有的输入流、输出流都可以分为字节（输入、输出）流、字符（输入、输出）流，处理字节的主要是（OutputStream/InputStream）系列，处理字符的，主要是（Reader/Write）系列。

2．常用的I/O流及功能

（1）以字节（Byte）为导向的输入流（InputStream 系列）。

- ByteArrayInputStream：把内存中的一个缓冲区作为 InputStream 使用。

- StringBufferInputStream：把一个 String 对象作为 InputStream。
- FileInputStream：把一个文件作为 InputStream，实现对文件的读取操作。
- PipedInputStream：实现了 pipe 的概念，主要在线程中使用。
- SequenceInputStream：把多个 InputStream 合并为一个 InputStream。

（2）以字节（Byte）为导向的输出流（OutputStream 系列）。

- ByteArrayOutputStream：把信息存入内存中的一个缓冲区中。
- FileOutputStream：把信息存入文件中。
- PipedOutputStream：实现了 pipe 的概念，主要在线程中使用。
- SequenceOutputStream：把多个 OutStream 合并为一个 OutStream。

（3）以 Unicode 字符为导向的输入流（Reader 系列）。

- CharArrayReader：与 ByteArrayInputStream 对应。
- StringReader：与 StringBufferInputStream 对应。
- FileReader：与 FileInputStream 对应。
- PipedReader：与 PipedInputStream 对应。

（4）以 Unicode 字符为导向的输出流（Write 系列）。

- CharArrayWrite：与 ByteArrayOutputStream 对应。
- StringWrite：无与之对应的以字节为导向的 stream。
- FileWrite：与 FileOutputStream 对应。
- PipedWrite：与 PipedOutputStream 对应。

（5）用于封装以字节为导向的，以下主要是用来修饰 InputStream 系列的各种输入。

- DataInputStream：从 stream 中读取基本类型（int、char 等）数据。
- BufferedInputStream：使用缓冲区。
- LineNumberInputStream：会记录 input stream 内的行数，然后可以调用 getLineNumber()和 setLineNumber(int)方法。
- PushbackInputStream：很少用到，一般用于编译器开发。

（6）用于封装以字符为导向的，主要是用来修饰 Reader 系列的各种输入。

- 没有与 DataInputStream 对应的类。除非在要使用 readLine()时改用 BufferedReader；否则使用 DataInputStream。
- BufferedReader：与 BufferedInputStream 对应。
- LineNumberReader：与 LineNumberInputStream 对应。
- PushBackReader：与 PushbackInputStream 对应。

（7）用于封装以字节为导向的，主要用来修饰 OutputStream 系列的各种输出。

- DataIOutStream：往 stream 中输出基本类型（int、char 等）数据。
- BufferedOutStream：使用缓冲区。
- PrintStream：产生格式化输出。

（8）用于封装以字符为导向的，主要用来修饰 Write 系列的各种输出。

- BufferedReader 与 BufferedWriter 对应，其中，BufferedReader 从字符输入流中读取文本，缓冲各个字符，从而实现字符、数组和行的高效读取，可以指定缓冲区的大小，或者可使用默认的大小。大多数情况下，默认值就足够大了，分别用来执行字符式的读和写的操作。而 BufferedWriter 则与之对应，用以实现文本写入字

符输出流，缓冲各个字符，从而提供单个字符、数组和字符串的高效写入。

- PrintWriter，用来向文本输出流打印对象的格式化表示形式。此类实现了 PrintStream 中的所有 print()方法。它不包含用于写入原始字节的方法。需要注意的是，PrintWriter 并没有与之对应的 PrintReader，因为 PrintWriter 主要是用来输出的。

（9）一个特殊的类：RandomAccessFile。

可通过 RandomAccessFile 对象完成对文件的读写操作，在产生一个对象时，可指明要打开的文件性质，r 表示只读；w 表示只写；rw 表示可读写；可以直接跳到文件中指定的位置。

12.3.6 Java GUI 编程

基于 P2P 的即时通信系统，需要提供用户一个可视的、友好的操作界面，这就需要用到 Java 的 GUI 编程知识。Java 的 GUI 编程（Graphic User Interface，图形用户接口），是在它的抽象窗口工具箱（Abstract Window Toolkit，AWT）上实现的，Java.awt 是 AWT 的工具类库，其中包括了丰富的图形、用户界面元件和布局管理器的支持。

1. Java GUI的基本知识

1）框架、监听器和事件

框架（Frame）是 Java 图形用户界面的基础，它就是我们通常所说的窗口，是 Windows/XWindow 应用程序的典型特征。Java 的图形用户界面是事件驱动的，并且由各种各样的监听器（Listener）负责捕捉各种事件。

如果需要对某一个组件的某种事件进行捕捉和处理时，就需要为其添加监听器。要在一个窗口（JFrame）激活时改变它的标题，就需要为这个窗口（JFrame 对象）添加一个可以监听到“激活窗口”这一事件的监听器——WindowListener。添加监听器通常由组件类提供的一个 addXXXXXListener()的方法来完成。比如 JFrame 就提供有 addWindowListener()方法添加窗口监听器（WindowListener）。

一个监听器常常不只监听一个事件，而是可以监听相关的多个事件。区分事件的方法靠重载监听器类（Class）的方法（Method）来实现，监听器监听到某个事件后，会自动调用相关的方法。只要重载这个方法，就可以处理相应的事件了。

在 JFrame 上发生的窗口事件（WindowEvent）包括：

- windowActivated(WindowEvent e)：窗口得到焦点时触发。
- windowClosed(WindowEvent e)：窗口关闭之后触发。
- windowClosing(WindowEvent e)：窗口关闭时触发。
- windowDeactivated(WindowEvent e)：窗口失去焦点时触发。
- windowDeiconified(WindowEvent e)：当撤销窗口图标化时触发。
- windowIconified(WindowEvent e)：当窗口图标化（图小化）时触发。
- windowOpened(WindowEvent e)：窗口打开之后触发。

2）按钮、切换按钮、复选按钮和单选按钮

按钮，Windows 窗口界面上的一个组件。切换按钮有两种状态的按钮，即按下状态和

弹起状态，若称为选择状态或未选择状态。复选按钮又叫复选框，用一个小方框中是否打勾来表示两种状态。单选按钮，又叫收音机按钮，以小圆框打点表示被选中。常成组出现，一组单选按钮中只有一个能被选中。

除一般按钮外，其余 3 种按钮都有两种状态，即选择（按下）状态和未选择（弹起）状态。切换按钮（JToggleButton）提供了一个 isSelected()方法用来判断当前所处的状态，返回值为真（true）时表示它处于选择状态，返回值为假（false）时表示它处于未选择状态。

复选按钮（JCheckBox）和单选按钮（JRadioButton）都是从 JToggleButton 继承的，所以也具有 isSelected()方法。单选按钮的特点决定了它们必须成组出现，而且一组中只能有一个能被选中。这用类 ButtonGroup 来管理。添加到 ButtonGroup 的多个单选按钮中，如果有一个被选中，同组中的其他单选按钮都会自动改变其状态为未选择状态。

3）文本输入框、密码输入框

文本输入框包括两种，单行文本输入框（JTextField）和多行文本输入框（JTextArea）。密码输入框则只有一种（JPasswordField）。JPasswordField 是 JTextField 的子类，它们的主要区别是 JPasswordField 不会显示出用户输入的东西，而只会显示出程序员设定的一个固定字符，如‘*’。

JTextField 有 5 个构造方法，其中常用的有 4 个，JTextField()、JTextField(int columns)、JTextField(String text)、JTextField(String text，int columns)等。其中，参数 text 是单行文本框的初始内容，而 columns 指定了单行文本框的宽度，以字符为单位。JTextField 中的文本内容可以用 getText()方法获得。也可以用 setText()方法指定 JTextField 中的文本内容。

JPasswordField 是 JTextField 的子类，其构造方法也是类似的。JPasswordField 提供了 setEchoChar(char ch)方法设置为了隐藏密码而显示的字符，默认为‘*’字符，上例中则设置为了‘#’字符（pwdField.setEchoChar('#');）。与 JTextField 一样，JPasswordField 也用 getText()方法和 setText()获得或者设置文本内容（当然在用户界面上是隐藏的）。

JTextField 是单行文本框，不能显示多行文本，如果想要显示多行文本，就只好使用多行文本框 JTextArea 了。JTextArea 有 6 个构造方法，常用的也是 4 个，JTextArea()、JTextArea(int rows，int columns)、JTextArea(String text)和 JTextArea(String text，int rows，int columns)，text 为 JTextArea 的初始化文本内容；rows 为 JTextArea 的高度，以行为单位；columns 为 JTextArea 的宽度，以字符为单位。如上例中就构造了一个高 5 行，宽 15 个字符的多行文本框（textArea = new JTextArea(5，15);）。

多行文本框默认是不会自动折行的（可以输入回车符换行），可以使用 JTextArea 的 setLineWrap()方法设置是否允许自动折行。多行文本框里文本内容的获得和设置，同样可以使用 getText()和 setText()两个方法来完成。

4）窗格、滚动窗格和布局管理

窗格（JPanel）和滚动窗格（JScrollPane）在图形用户界面设计中大量用于各种组件窗口上的布局（Layout），布局由布局管理器（Layout Manager）来管理。常用的布局管理器有 FlowLayout、BorderLayout、GridLayout、BoxLayout 等，其中 FlowLayout 和 BorderLayout 最常用，如表 12.1 说明了它们的布局特点。

表 12.1　常用布局管理器的分类及特点

类　　名	功 能 说 明
FlowLayout	将组件按从左到右从上到下的顺序依次排列，一行不能放完则折到下一行继续放置
BorderLayout	将组件按东、南、西、北、中 5 个区域放置，每个方向最多只能放置一个组件
GridLayout	形似一个无框线的表格，每个单元格中放一个组件
BoxLayout	就像整齐放置的一行或者一列盒子，每个盒子中一个组件

任何布局管理器都需要用在容器上，如 JFrame 的 Content Pane 和 JPanel 都是容器。容器组件提供了一个 setLayout()方法，就是用来改变其布局管理器的。默认情况下，JFrame 的 Content Pane 使用的是 BorderLayout，而 JPanel 使用的是 FlowLayout。但不管怎样，都可以调用它们的 setLayout()方法来改变其布局管理器。

滚动窗格是一个能够自己产生滚动条的容器，通常只包容一个组件，并且根据这个组件的大小自动产生滚动条。

2．Java中的GUI实现方式

采用 AWT（抽象窗口工具集）从而可使 GUI 适用于不同 OS 的环境。它有两个特点：

- ❑ 其具体实现由目标平台下的 OS 来解释，从而导致 Java GUI 在不同平台下会出现不同的运行效果（窗口外观、字体等的显示效果会发生变化）。
- ❑ 组件在设计时不应采用绝对定位，而应采用布局管理器来实现相对定位，以达到与平台及设备无关。

3．新增的Swing GUI组件

AWT 组件以及事件响应不及微软的 SDK 丰富（因为有些 OS 平台无微软的 Windows 组件），Sun 在 Java 2 中新增了 Swing GUI 组件。但是，AWT 比较简单，功能也能满足大多数的界面需求，特别在 Java Applet 的设计中受到了普遍的应用。同时，这个讨论也为进一步研究 Swing GUI 组件打下了比较扎实的基础。

以上就是 Java GUI 编程的基本知识，这些基本组件包括动作、监听器等知识都会在系统中用到，理解以上知识对系统的实现很有意义。但要熟练掌握 GUI 编程，还需要读者在不断的实践中学习。

12.4　系统的组织与基本类的设计

在 12.3 节中重点地讲解了本系统的实现机制，对系统开发的原理及用到的知识点也进行了说明，有了设计的目标和需求，也有了系统实现的理论基础，那么就可以从整体上对系统进行组织与设计了。本节就重点讲解一下系统的组织与基本类的设计。

12.4.1　系统执行流程

要对系统进行整体上的组织与设计，就要对系统的整个执行流程有清晰的理解，就像盖一栋高楼一样，在正式建造楼房之前，建设者应该对建造的顺序和过程了然于胸，什么

时候打地基、什么时候布线、什么时候装修等，应该有严格的执行顺序。构建一个程序也是一样，只有清晰地设定程序的执行流程才能有效地兼顾各个细节，统一协调各个模块，这样才能保证程序的健壮，不至于漏洞百出。

所以，在系统的组织设计以前，先分析一下程序的执行流程。

（1）系统启动，进入步骤（2）。

（2）启动用户登录/注册界面，如果用户登录一个已有的频道则执行步骤（3），如果用户注册一个新频道则执行步骤（5），否则执行步骤（22）。

（3）刷新当前频道列表，如果频道列表没有频道信息则执行步骤（5），否则执行步骤（4）。

（4）从频道列表选择一个要加入的频道，输入此频道的认证密码，如果密码认证正确则执行步骤（6），否则执行步骤（22）。

（5）输入一个新频道的名称和新频道的注册密码，注册一个新的频道，单击“确定”按钮后执行步骤（6）。

（6）进入频道主窗口，如果在频道主窗口的 Peer 列表中显示有其他 Peer 的信息，就可以执行步骤（7），否则执行步骤（20）。

（7）此频道中有 1 个以上的 Peer，就可以实现 Peer 间信息的交互了，具体的交互过程是步骤（8）～步骤（19）的反复执行过程。直到出现步骤（21），交互过程结束。

（8）共享文件目录，单击“共享目录”按钮，可以将本地目录共享给本频道中的其他所有 Peer。

（9）发送文件，单击“发送文件”按钮，可以将自己的文件发送给此频道中的其他所有 Peer。

（10）接收文件，单击“接收文件”按钮，弹出接收文件对话框，在此对话框中可以执行步骤（16）的操作。

（11）建立私人会话，从 Peer 列表中选择一个 Peer，右击，在弹出的快捷菜单中选择“私人聊天”选项，会弹出私人会话的聊天界面，在此界面中可以执行步骤（19）。

（12）查看共享，从 Peer 列表中选择一个 Peer，右击，在弹出的快捷菜单中选择“显示共享”选项，如果此 Peer 有共享文件，则执行步骤（14），否则执行步骤（15）。

（13）清屏，单击“清屏”按钮，可以将频道主界面中消息窗口里的内容清空。

（14）弹出此 Peer 的文件共享界面，可以查看共享的文件列表，在此界面中可以执行步骤（17）和步骤（18）的操作。

（15）弹出提示信息，会提示当前 Peer 没有共享文件，然后显示文件列表为空的共享界面。

（16）在接收文件的弹出界面中可以查看接收的文件列表，在此界面中可以执行步骤（17）和步骤（18）的操作。

（17）单击文件列表中的某一文件名，可以在界面的另一区域显示此文件的内容。

（18）右击文件列表中的某一文件，可将此文件存储到本地。

（19）两个 Peer 间可以在私人聊天界面中进行 Peer 间的会话。

（20）等待其他 Peer 加入此频道。

（21）所有 Peer 退出，频道关闭，执行步骤（22）。

（22）程序退出。

以上描述的是整个系统详细的执行流程，为方便读者理解，在这个流程的基础上进行简化就形成了如图 12.10 所示的执行流程图。这个流程图可以大致地反映基于 P2P 即时通信系统的整个执行的先后顺序和交互过程。

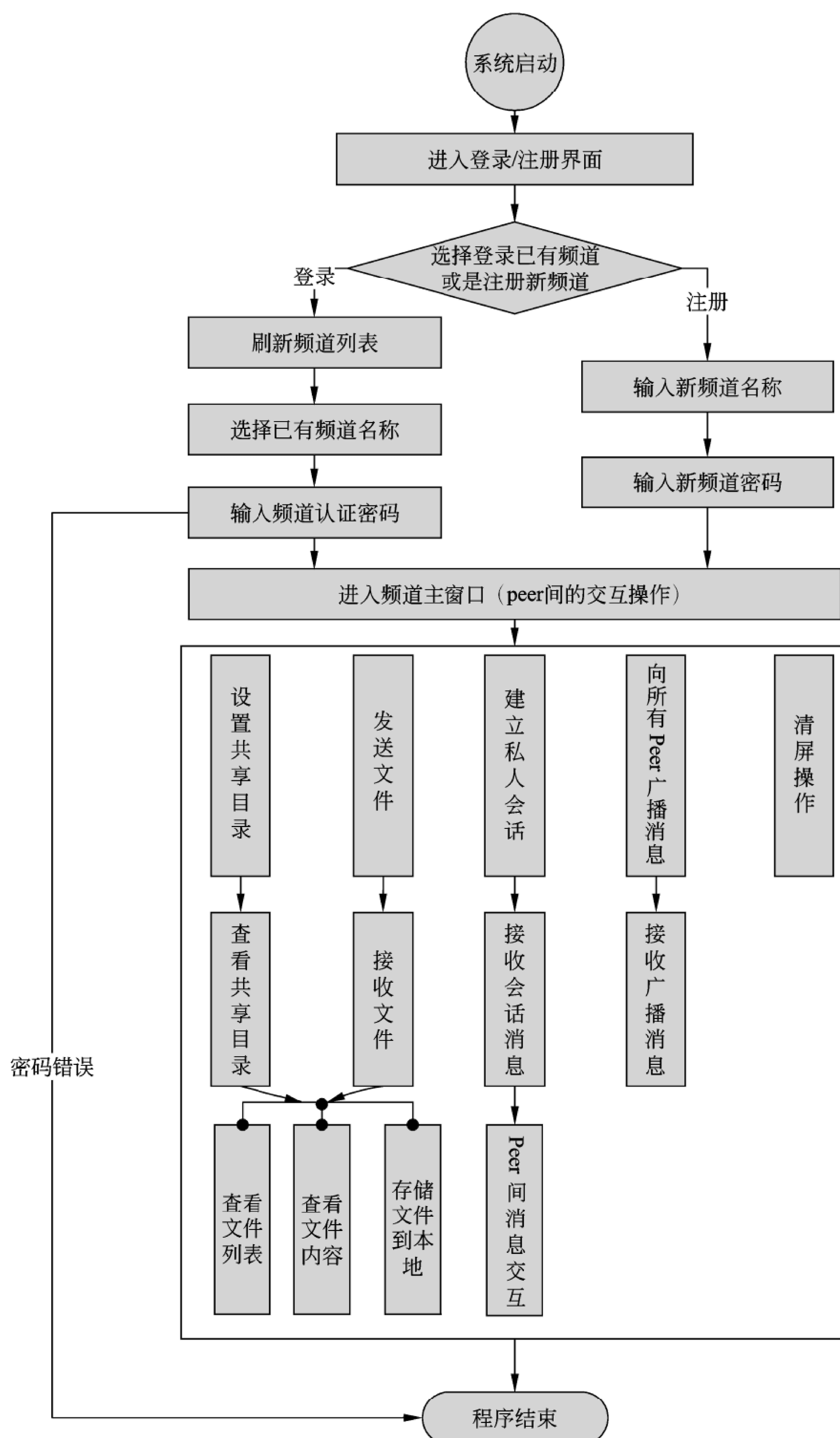


图 12.10　P2P 即时通信系统执行流程图

12.4.2　模块划分与基本类的组织

在本章 12.2 节里已经对系统规划有了详细说明，将整个系统分为 4 个大的模块，根据这些模块的特征、每个模块要完成的功能，就可以针对不同的模块进行类的组织与功能的定义了。下面就分析一下，每个模块由哪些基本类构成，这些类是如何组织的。

1．界面显示模块（UI）

界面显示模块主要用于系统运行过程中所有界面的显示，根据需求分析的说明，主要由以下几个类组成，这些类的组织方式、类的命名及功能说明如表 12.2 所示。

表 12.2　界面显示模块的类设计及功能描述

组 织 方 式	类　　名	功 能 说 明
Package p2p.chat.ui	chatStart	系统运行的开始界面，显示用户登录一个已有的频道或是注册一个新的频道
	mainFrame	系统的主界面，显示 Peer 列表、会话消息和相应的菜单工具条
	privateConvFrame	两个 Peer 间的私人聊天界面，有一个基本的消息输入框和会话显示框
	recvFilesFrame	用户接收文件的界面，此界面可以显示文件的内容，也可以利用右键操作将文件存储到本地
	userShareFrame	Peer 结点的共享文件显示界面，可显示共享目录的结构及文件内容
	FileDialog	文件选择对话框，用于供用户浏览本地目录，选择所需的文件

2．功能模块（Function）

在需要分析中已经说过，本系统除了系统主界面所展示的基本功能外，不需要另外两个独立的子功能，分别为处理文件共享、处理 Peer 间的私人会话。那么功能模块里就对应着两个类，此模块中类的组织方式、类的命名及功能说明如表 12.3 所示。

表 12.3　基本功能模块的类设计及功能说明

组 织 方 式	类　　名	功 能 说 明
Package 2p.chat.fun	FileSharing	用于处理文件共享的功能，主要是查看其他 Peer 结点共享的文件列表、显示文件内容、将共享文件下载到本地等
	PrivateConversation	用于处理两个 Peer 间的私人会话，有消息输入框和内容显示，当有新消息到来时，会自动弹出对话框

3．消息模块（Message）

本系统中所有通过网络交互的消息都由消息模块来完成，根据系统的需求，它有多个不同的消息类型，消息模块中类的组织方式、类的命名及功能说明如表 12.4 所示。

表 12.4　消息模块的类说明及功能描述

组 织 方 式	类　　名	功 能 说 明
Package p2p.chat.message	ChannelMessage	频道消息，是由 Peer 结点发送到 P2P 频道的文本消息
	PrivateMessage	私人会话消息，由一个 Peer 结点发送到另一个 Peer 结点的文本消息
	SharingListMessage	共享列表消息，用来告诉某一 P2P 频道中的每一个结点，当前用户的文件共享情况
	AttachmentMessage	附件消息，代表了用户将文件发送到网络中的消息，也包括显示文件的内容
	PeeerMessage	结点消息，用来显示和表明用户结点的一些信息
	ServiceMessage	服务消息，由 P2P 频道用于同步和管理的消息

4．网络模块（Network）

网络模块主要用于实现底层的 P2P 机制，确切地说是用于发现网络中的 Peer 结点，根据需求的规定和实现机制的分析，这一模块主要由 3 个类完成，这些类的组织方式、类的命名及功能说明如表 12.5 所示。

表 12.5　网络模块的类说明及功能描述

组 织 方 式	类　　名	功 能 说 明
Package p2p.chat.net	NetworkDispatcher	网类分发，构建一个网络分发型，主要用于处理数据的加密、解密及向所有结点分发消息
	MulticastDispatcher	多点分发，网络分发的具体实现，主要通过多点 Sockets 在网络中发送数据
	ChanAdvertiserThread	频道广告线程，一个定时向网络中广告当前频道信息的类，多线程执行，不停地循环

5．其他实体类的说明（Other）

除了以下各个模块以外，最基本的应该有两个实体，一个是用户，另一个就是 P2P 频道。用户，也就是所说的 Peer 结点，它是系统的参与者，而 P2P 频道，它是一个模拟的 P2P 网络，Peer 的所有行为都在这个 P2P 频道中完成。另外，从程序的设计来考虑，还要有一个主类，就是 main class，它是系统的执行入口，所以还应该有以下 3 个类，它们的组织方式、类的命名及功能说明如表 12.6 所示。

表 12.6　其他实体类说明及功能描述

组 织 方 式	类　　名	功 能 说 明
Package p2p.chat.net	Peer	结点类，作为一个实体用来描述 Peer 结点的各种信息
	Channel	频道类，通过一个个频道来模拟 P2P 网络，它所描述的是 P2P 频道的基本信息，如名称、密码、创建者等
	Main	系统的主类，包括一个主函数，系统执行的入口

以上是对系统模块的初步划分以及相关类的说明，在具体开发的过程中，根据面向对象、设计模式等要求，还会加入其他的类和关系，这些在后文类的设计里会讲到。

12.4.3　面向对象的设计与类的构建

依照需求分析和设计目标的要求，上文已经对每个模块的划分与类的组织进行了详细的说明。在上文的分析中，虽然每一个类都有一个名字，也规定了其相应的功能，但就整个系统而言，它们都是一个个独立的模块、一个个松散的类，没有任何关系，这是无法形成一个有组织的系统的。本节就具体讲，如何按照面向对象的设计思想来设计各个模块及相关类之间的关系。

从整个系统来看，基于 P2P 的聊天系统有 4 个大的对象，第 1 个就是用户对象，也就是 Peer，它是整个系统的参与者。第 2 个是频道对象，它是 Peer 间交互的虚拟平台，第 3 个是 Message 对象，它是整个系统所有交互的消息、数据的抽象，第 4 个就是网络层对象，用于多点广播数据以发现结点，并实现结点间的通信。下面就分别讲一下这 4 个对象之间的关系。

1. Peer对象

Peer 是 P2P 聊天系统的主要参与者，与 Peer 有关的类，如图 12.11 所示。

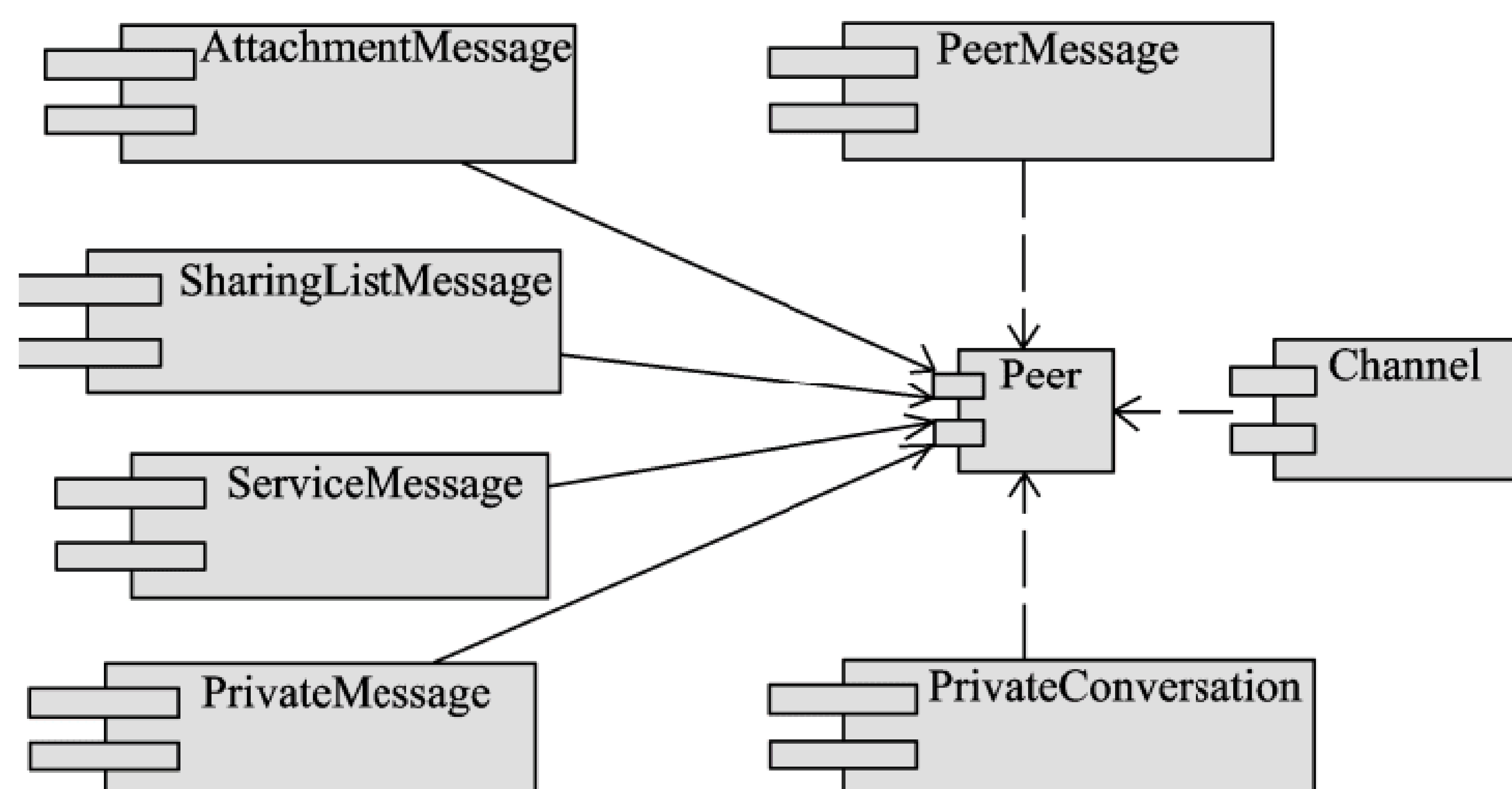


图 12.11　Peer 对象与其他类的关系图

在图 12.11 所示的组件中，PeerMessage 类是 Peer 间交互的消息类，Channel 是 Peer 活动的平台，提供 Peer 聊天的基本功能，PrivateMessage、PrivateConversation 是与 Peer 间进行私人聊天有关的类。ServiceMessage 主要用于 Peer 间消息的同步和管理，如 Peer 结点的加入、退出，就需要此消息进行同步的向其他 Peer 结点进行说明。

2. Channel对象

Channel 对象是一个虚拟的 P2P 平台，同一 Channel 下的所有结点可以对等进行消息会话、文件交互和资源共享，与 Channel 有关的类，如图 12.12 所示。

图 12.12 所示的组件图中，ChannelMessage 类，是由 Peer 结点发向频道的消息，而 ChanAdvertiserThread，是一个频道消息的广告类，当一个 P2P 频道创建完成后，由 ChanAdvertiserThread 类向网络中不停的广告当前的频道信息，当有新的 Peer 结点加入时，收到这个频道广告消息后，就会更新自身的频道列表。

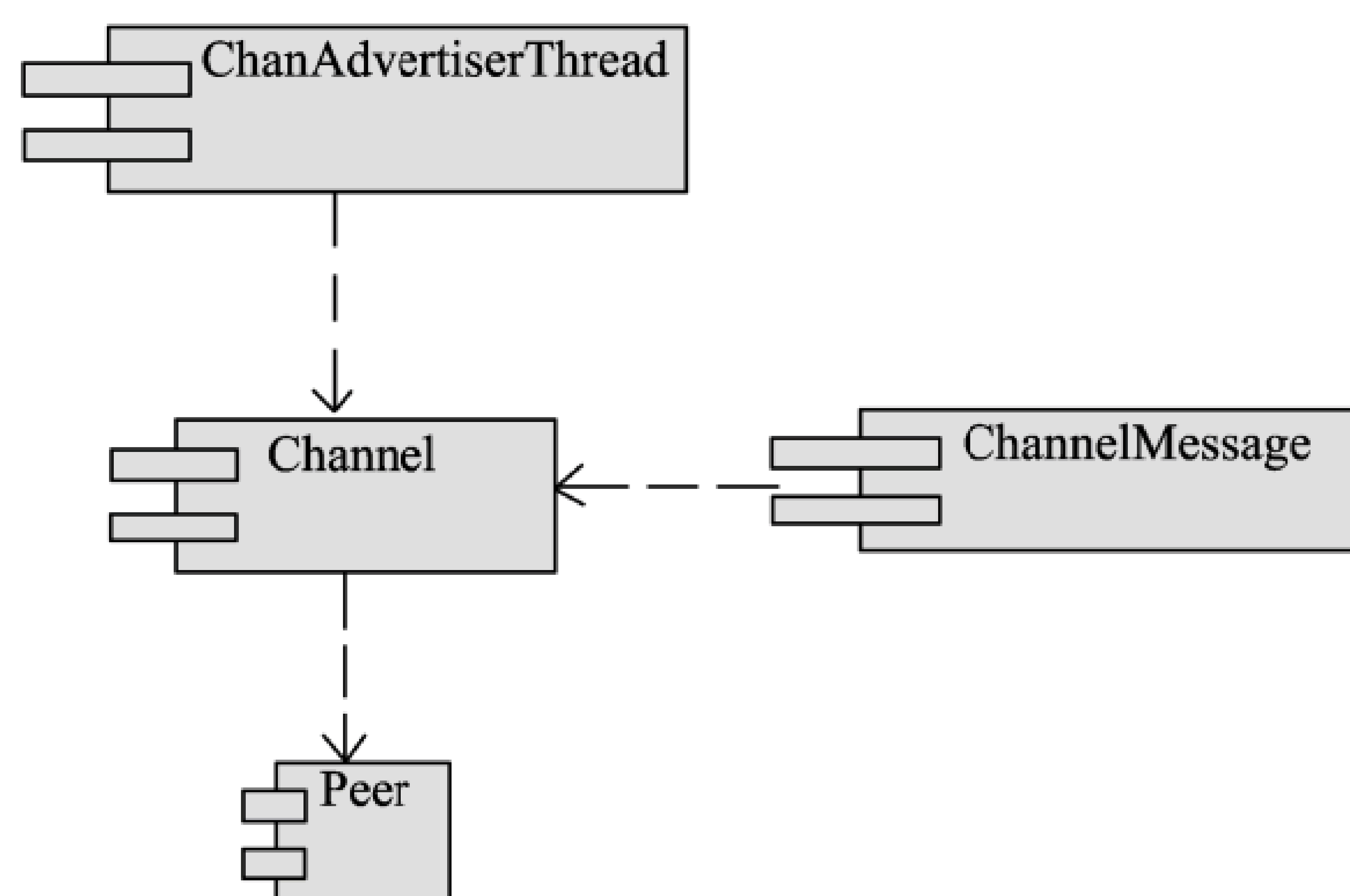


图 12.12　Channel 对象与其他类的关系图

3．Message对象

在消息模块里已经讲过，所有的通过网络发送的东西都表示为消息，所以本系统中消息有很多种。从面向对象的角度来说，这些消息必然有共同的地方，所以，设计一个 Message 类作为父类，其他的所有消息都继承自 Message。Message 父类与其子类间的继承关系如图 12.13 所示。

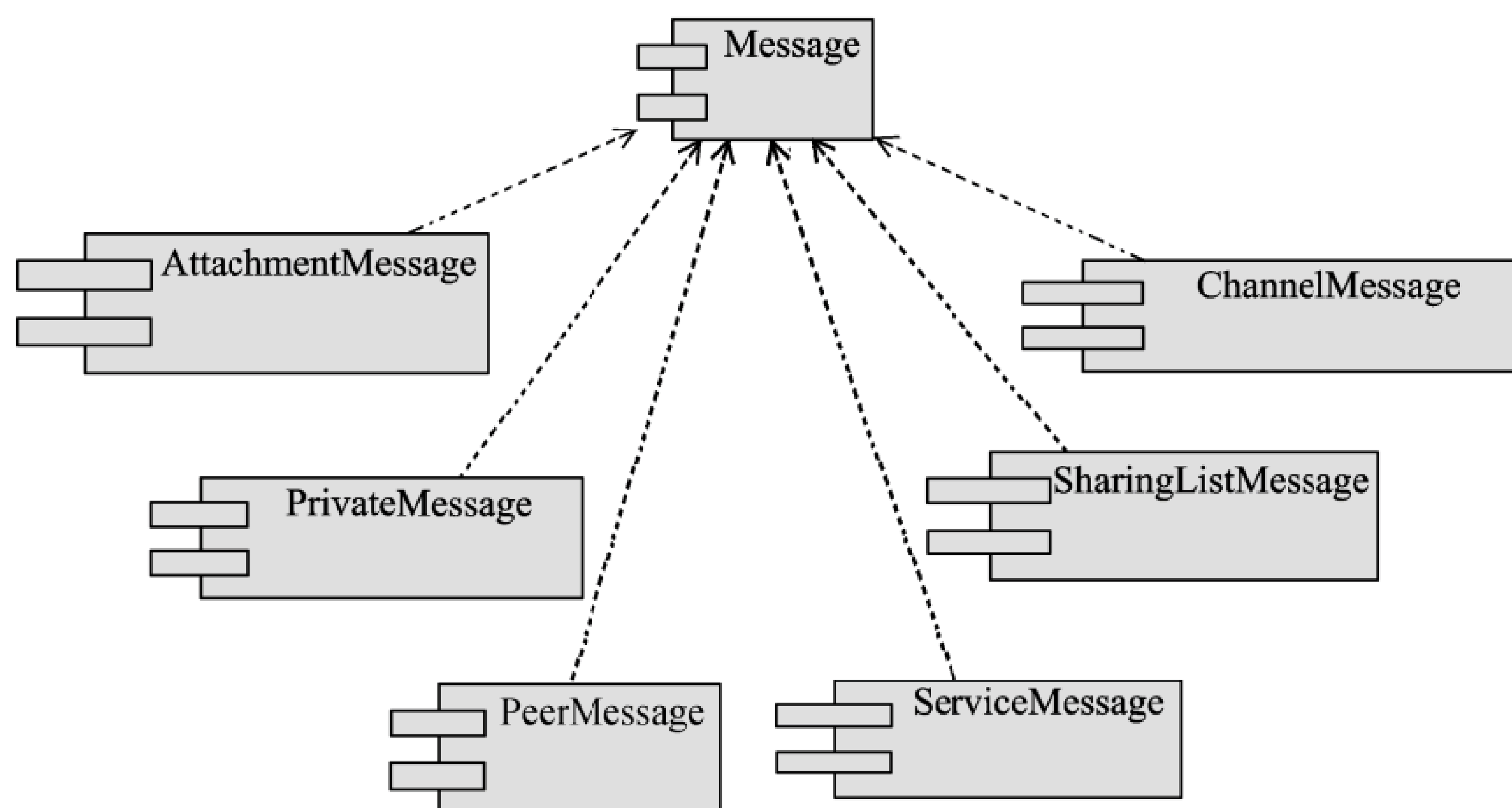


图 12.13　Message 父类与其子类之间的继承关系图

如图 12.13 所示，由父 Message 派生出 6 个消息类，分别为 PrivateMessge、PeerMessage、AttachmentMessage、ChannelMessage、ServiceMessage 和 SharingListMessage。它们分别应用在不同的功能上，比如 ChannelMessage，它主要描述的是 Peer 结点发向 P2P 频道的消息，那么，首先要得到这个频道的名称，通过 GetChannel()方法可以得到。其次，要有向频道写入和读取对象的方法，所以有 writeObject()和 readObject()两个方法，通过这几个简要的方法就可实现 ChannelMessage 的功能。

4．网络对象

网络对象主要提供一种 Peer 结点的发现机制，并实现结点间的通信，在上文的实现机

制里已经讲过，主要是通过多点广播来实现的。所以网络对象只有两个类，它们之间的关系如图 12.14 所示。

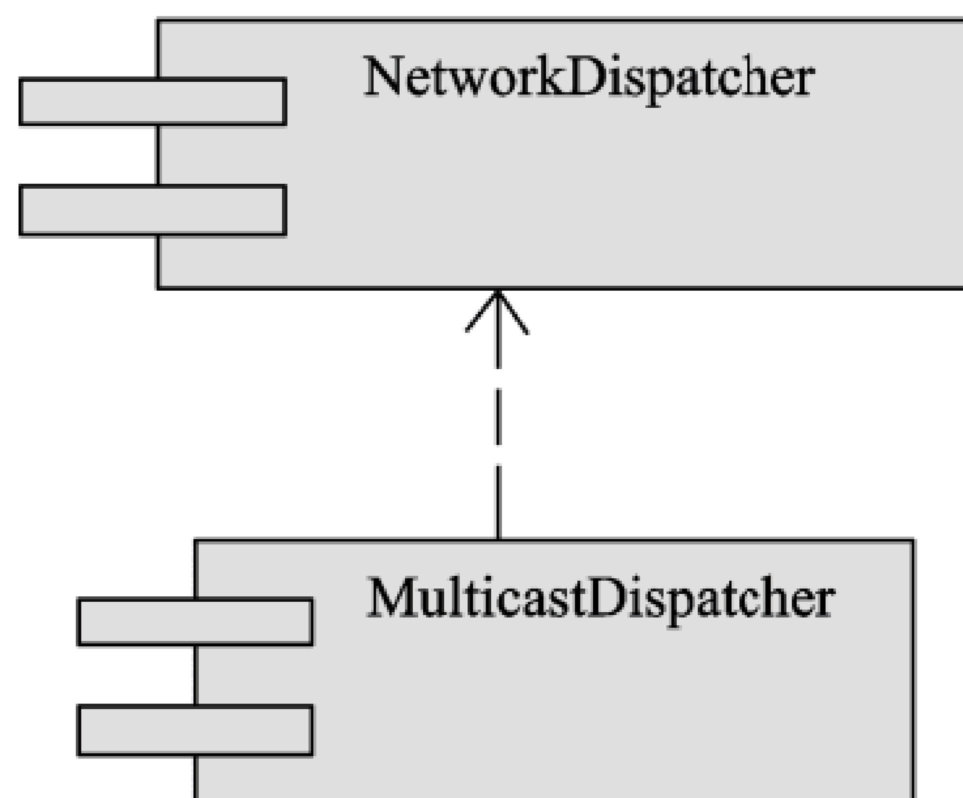


图 12.14　网络对象模块两个类之间的关系

图 12.14 中，NetWorkDispatcher 类与 MulticastDispatcher 类是继承关系，后者继承前者，NetWorkDispatcher 类主要用来实现消息的发送和消息的加密。MulticastDispatcher 类则是通过 Java 中的多点 Sockets 来具体实现消息的多点发送。

5．统一管理对象

上文中讲解了各个类的作用，也讲了类与类之间、模块之间的相互关系。就整个系统的开发而言，还需一个统一的管理对象来对这些类进行集中的调度和管理，这个管理对象就是 Manager 类。Manager 类是一个单例模式的类，这种单例模式能确保 Manager 类只有一个实例，而且自行实例化并向整个系统提供这个实例，它集中管理系统中的各个类的调度和执行。

Manager 类与其他组件模块之间的关系如图 12.15 所示。

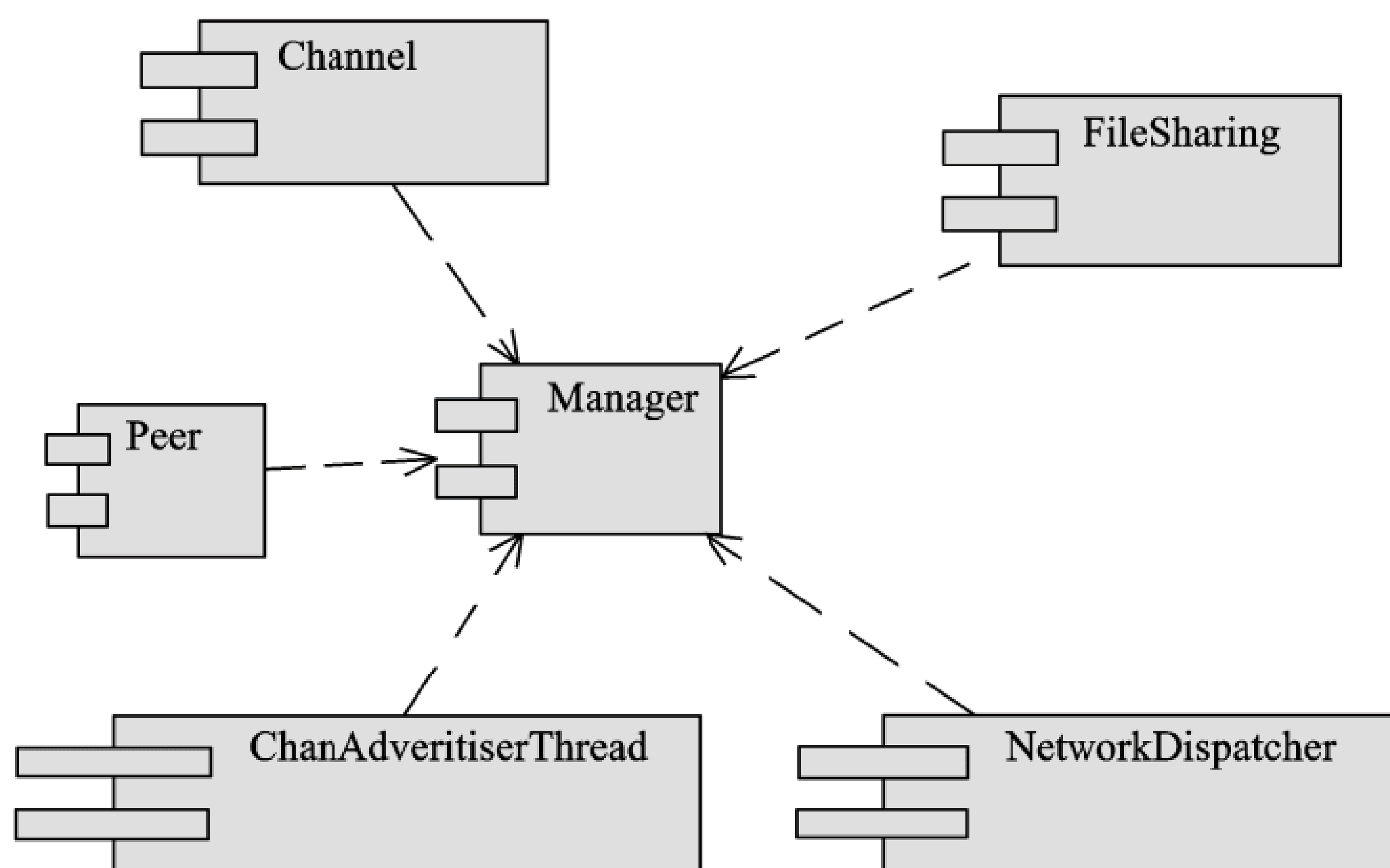


图 12.15　Manager 管理类与其他组件模块之间的关系

由图 12.15 可以看出，Manager 基本上管理了系统几个主要的模块，这样，Manager 通过管理这几个核心模块就管理了整个系统的调度，关于 Manager 类的详细信息会在后文的开发实例中具体讲解。

注意： 单例模式是设计模式的一种，它有 3 个基本要点：一是某个类只能有一个实例；二是它必须自行创建这个实例；三是它必须自行向整个系统提供这个实例。本系统中的 Manager 类，类似于一个集中的资源管理器，所以设计成单例模式。

12.4.4　系统的全局关系结构

由以上各个模块的分析，再加上 Manager 类的统一管理，将它们整合起来以后，整个系统的全局关系结构也就清楚了。如图 12.16 所示，就是本系统的全局关系结构图。

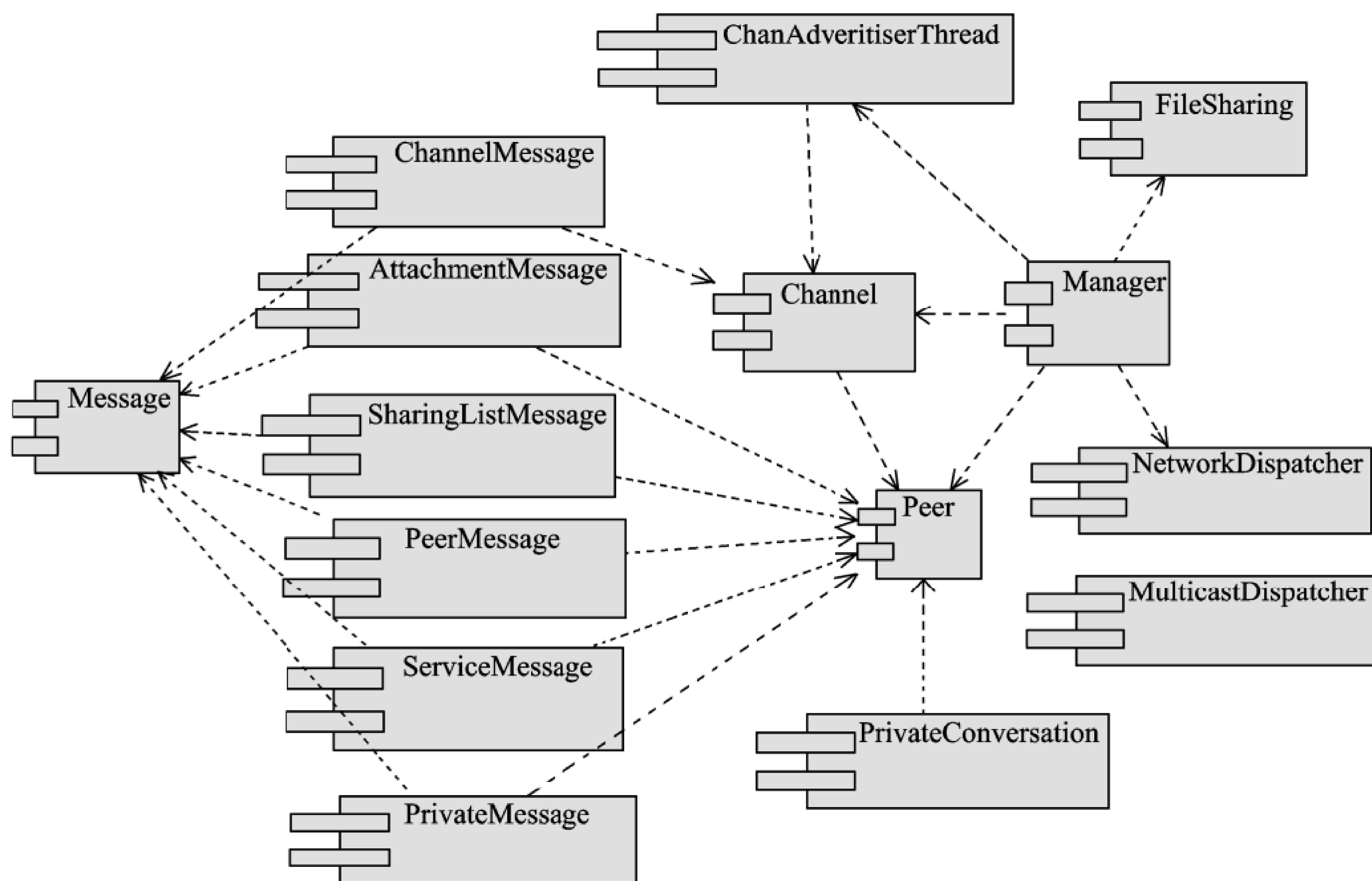


图 12.16　P2P 即时通信系统的全局关系结构图

由 12.16 就可以清楚地看出，整个系统 Peer 对象是核心，因为整个聊天系统就服务于 Peer 间通信的系统。Channel 对象是 Peer 平台，模块间的关系及调度由 Manager 类统一管理，整个系统的结构就显得很清晰，模块间的功能划分、调用关系也一目了然。

关于 P2P 即时通信系统的模块划分与整体设计就讲到这里，从第 13 章开始，就依照这样一个结构来开发实现一个基于 P2P 的即时通信系统。

12.5　系统的开发及编程实现

本章的前 4 节对 P2P 即时通信系统从设计到分析都进行了详细的讲解，对实现机制和关键技术也进行了深入的分析，对类的命名及组织方式也进行了说明，有了这些知识做基础，就可以进行正式的系统开发和编程实现了，本节将带领读者一起开发一个基于 P2P 的即时通信系统。

12.5.1　搭建工程框架

开发系统之前，首先要把工程框架搭好，这个框架就是 Eclipse 开发工具下的 Java 工程框架。

1．新建工程

打开 Eclipse 的开发平台，根据 Eclipse 新建工程的方法，选择 file | new | Project 选项，新建一个 Java Project，命名为 p2pchat，工程建成后的效果如图 12.17 所示。

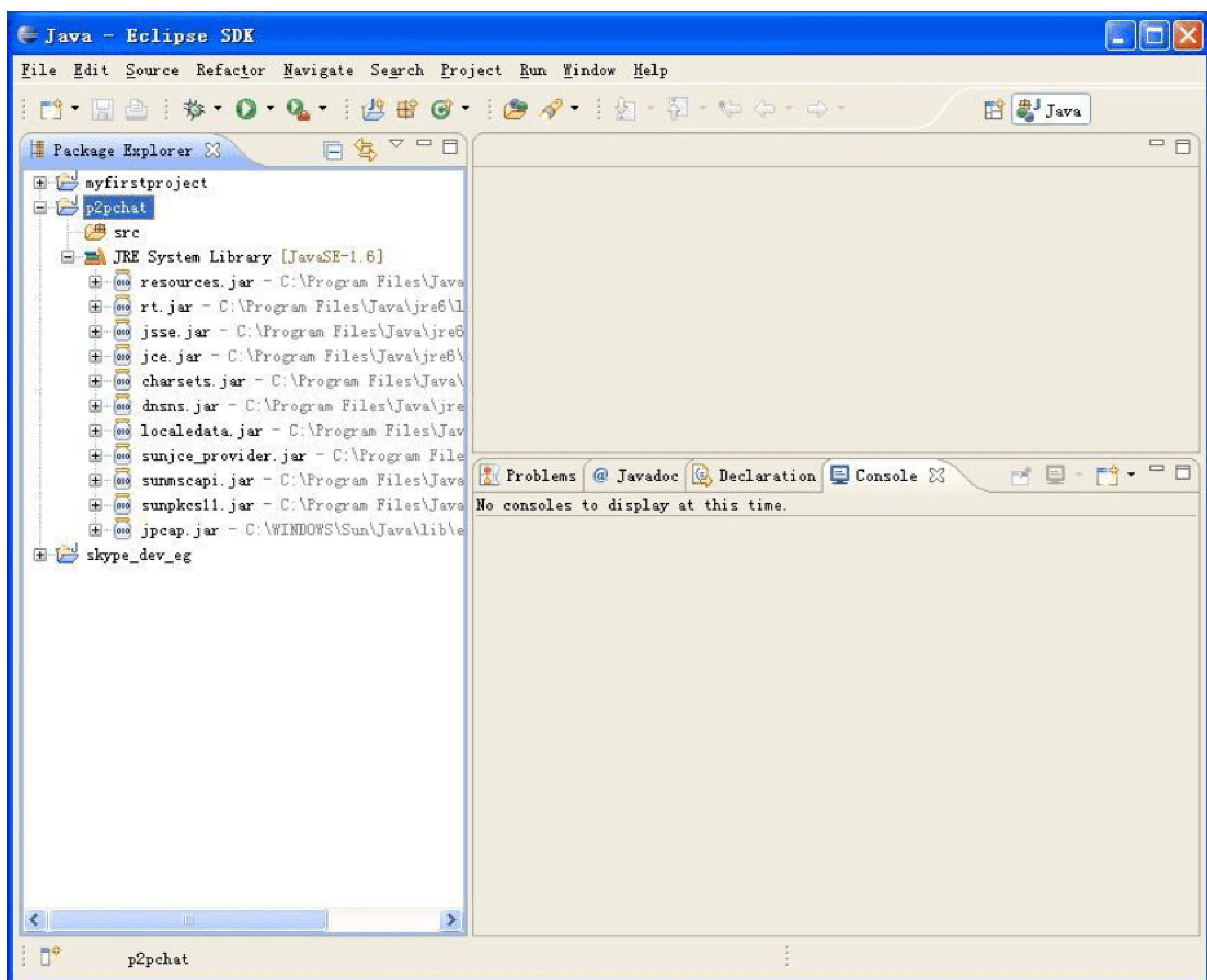


图 12.17　p2pchat 系统的工程框架图

2．建包

根据上文所讲的对 P2P 即时通信系统的模块划分，为更好地组织代码，就以模块为分类在 Eclipse 中新建相应的包，每一个包对应一个模块，所以要新建 4 个包，分别为：

- p2p.chat.net（网络模块）。
- p2p.chat.message（消息模块）。
- p2p.chat.fun（功能模块）。
- p2p.chat.ui（界面模块）。

其中，在 p2p.chat 下存放主程序和其他的实体类，另外，还有一个 p2p.chat.test 包，用于存放一些必要的测试程序，如上文讲到的多播消息发送和接收的测试程序就放在这个 test 包里。建成后的包及 p2pchat 工程结构如图 12.18 所示。

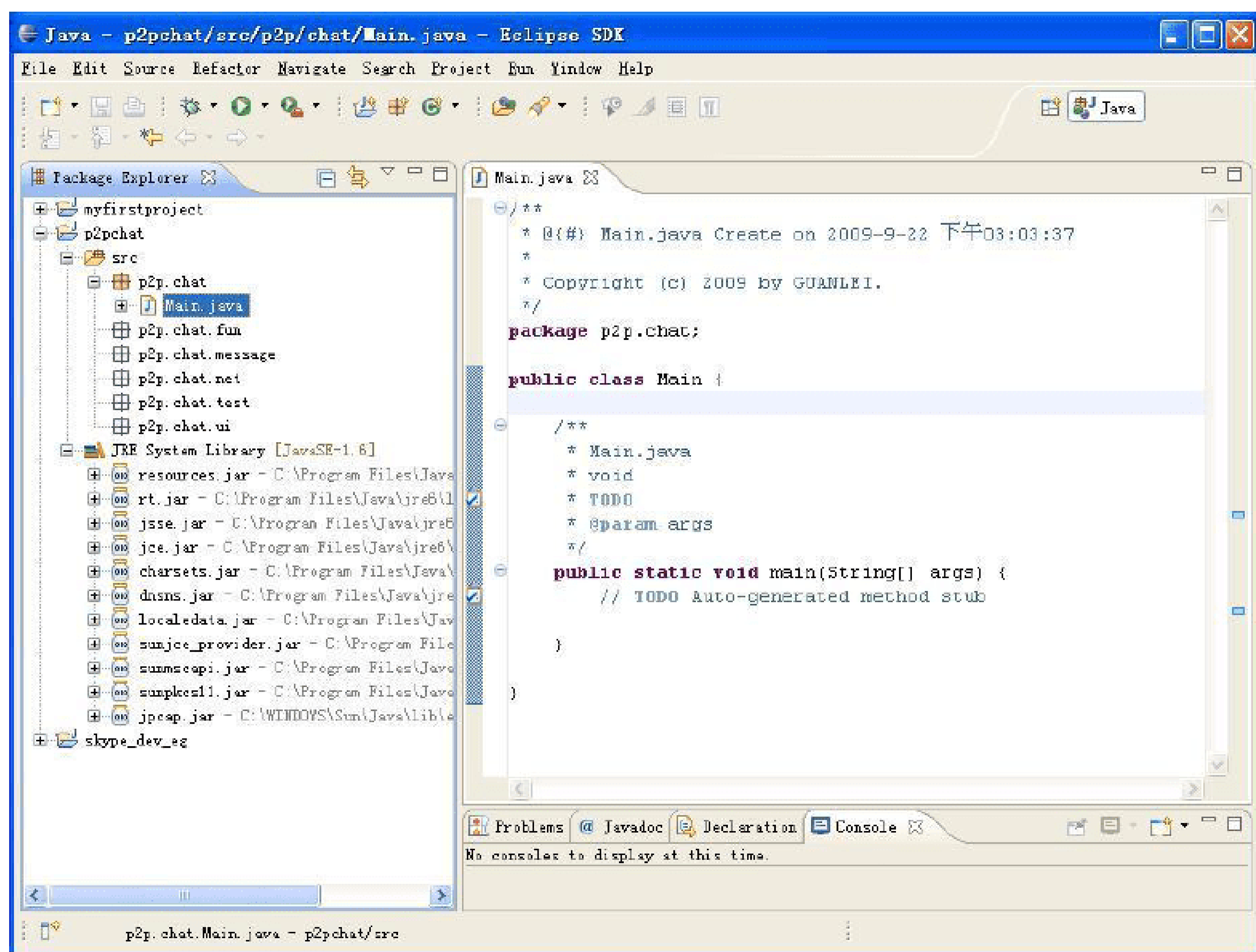


图 12.18　p2pchat 系统的包结构图

图 12.18 中显示了 p2pchat 工程的目录结构，并新建了一个内容为空的 Main 类，此类也是本系统的入口方法类，在整个系统开发完成后，再将此类补充完整即可。

从下面的内容开始，将带领读者学习类的设计和系统开发方法，读者需要注意的是，本文是按照模块化的设计思想来讲解开发过程的，与实际的开发顺序不同。模块之间的类会涉及相互引用的问题，也就是在讲解第一个类的时候，会出现还未讲解的对第二个类的引用，在这些问题在文中会有相应的注释说明。其实只要能明白系统的全局结构，在完成整个系统的开发后，类之间的调用关系也就十分清楚了。

12.5.2　网络模块的实现

网络模块是最底层的模块，主要用于实现 P2P 网络模型，本系统中主要通过 IP 多播机制来实现这一模型，就 P2P 网络而言，它最核心的要求是“结点对等”，没有服务器的概念，要想在没有中心服务器的情况下找到网络中的其他对等点，那么对等点就需要定期使用 IP 多播的方式来宣布自己的存在。

IP 多播，也叫多点广播，多播的消息里包含了主机名和一个用于正常通信的端口，当对此消息感兴趣的其他对等点检测这个消息后，就会抽取出主机名和端口号，并通过主机和端口消息与此结点建立一个通信通道，这样两个 Peer 之间就可以建立连接并进行通信了。

结点之间在网络中传输数据时是需要加密的，数据的加密主要用到 Java 的加密编程技术，这部分的内容在本章第 12.3 节的实现机制里已有说明，这里就不再赘述。

在网络模块里主要有两个类，它们之间是继承关系，其中 NetworkDispatcher 类为父类，主要定义了向网络中发送和接收数据的基本方法，MulticastDispatcher 类是子类，此类通过 Java 中的多点 Sockets 机制具体实现了数据的发送和接收。这两个类的关系及类的结构图如图 12.19 所示。

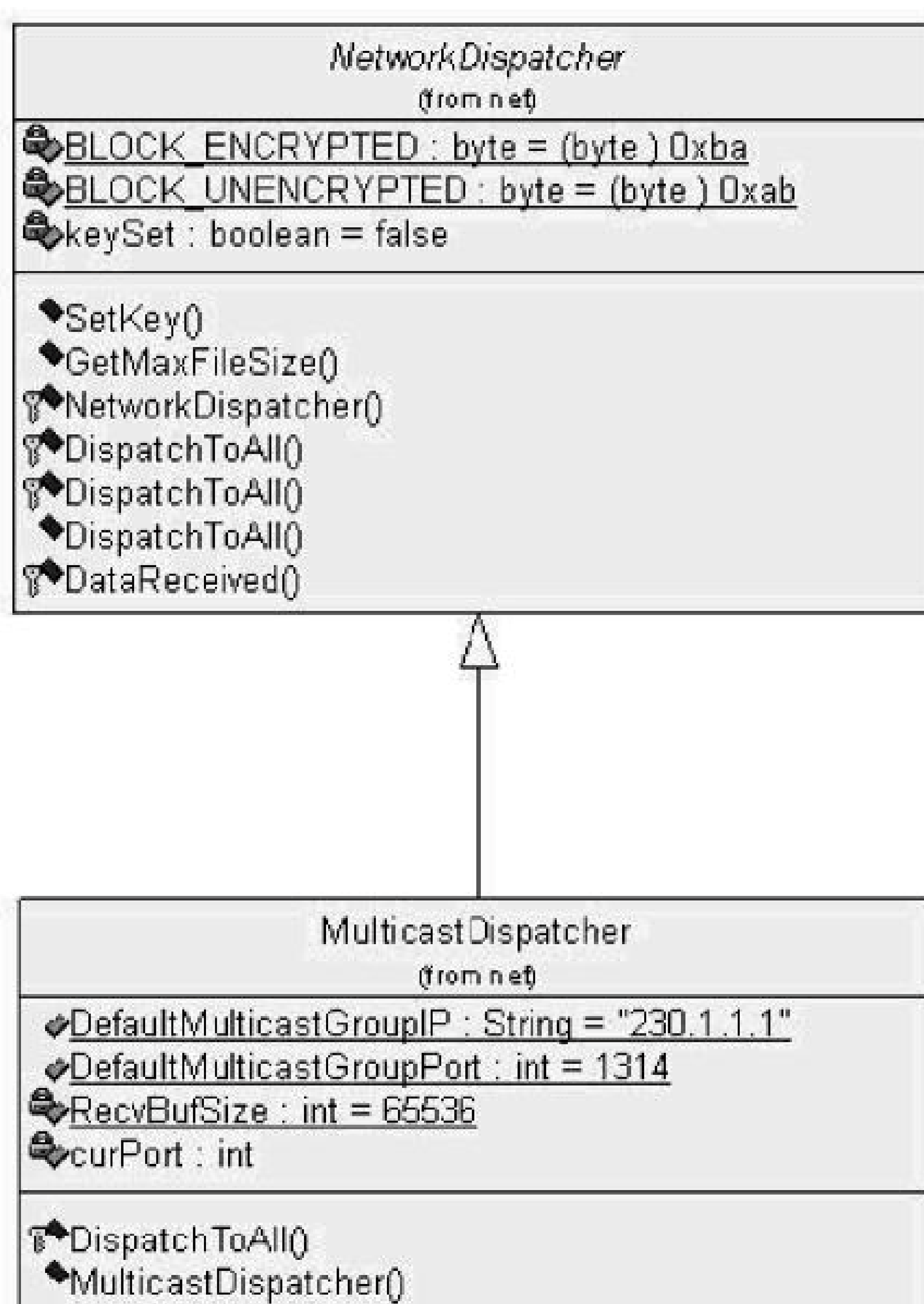


图 12.19　网络模块的类图及继承关系

图 12.19 所示的类图中，清楚地表明了两个类之间的继承关系以及它们主要的属性和方法。下面就分别讲一个这两个类的具体实现。

1．NetworkDispatcher类的实现

在 NetworkDispatcher 类中，它主要完成数据的加密和发送，有 4 个属性和 7 个方法，其中 SetKey()方法就是实现数据的加密，而 DispatchToAll()方法，很显然就是发送数据了。根据参数的不同执行不同的分发方式，因而也就有不同的构造方法。根据 NetworkDispatcher 的类图和系统对此类的要求，可以进行类的设计与开发了。

NetworkDispatcher 类位于 p2pchat 工程中的 p2p.chat.net 包下，以下是 NetworkDispatcher 类中主要方法的说明，详细完整的实现源代码请读者参考随书所附光盘。

【源代码示例：\ch12\ch12_code\p2pchat\src\p2p\chat\net\NetworkDispatcher.java】

NetworkDispatcher 是网络模块的重要组成部分，在整个即时通信系统中负责完成网络底层的数据分发工作，具体的实现中包括数据的加密、分发和接收等。

以下是 NetworkDispatcher 类的定义、变量的引用声明等。

```
/**
 *NetworkDispatcher 类，在本文的即时通信系统中，完成网络模块的功能，主要用来实现网络
 *底层的数据分发、数据加密、数据接收等功能。
 */
package p2p.chat.net;                                    //声明程序的包路径
//引入与程序实现相关的包，包括 I/O 操作、加密操作
import java.io.*;
import java.security.*;
import javax.crypto.*;
import p2p.chat.Manager;
import p2p.chat.message.Message;
import com.sun.crypto.provider.SunJCE;
```

```
//定义一个 abstract 的类，类名为 NetworkDispatcher，此类无法进行实例化
public abstract class NetworkDispatcher {
//定义一个 byte 型名为 BLOCK_ENCRYPTED 的静态常量，用来标识需要加密的数据
private static final byte BLOCK ENCRYPTED=(byte)0xba;
//定义一个 byte 型名为 BLOCK_ UNENCRYPTED 的静态常量，用来标识无须加密的数据
private static final byte BLOCK UNENCRYPTED=(byte)0xab;
//定义一个 boolean 类型的 keySet 变量，用来标识，是否设置过密码
private boolean keySet=false;
//定义一个 Cipher 对象，用来加密
private Cipher encCipher;
//定义一个 Cipher 对象，用来解密
private Cipher decCipher;
//定义一个 SecretKeySpec 对象
SecretKeySpec keyIV;
// 声明一个 PBEParameterSpec 对象
private PBEParameterSpec paramSpec;
//定义一个 SecretKey 对象
private SecretKey secretKey;
//定义一个字节类型的随机串，在以下的加密算法中会用到
private static byte[] salt = {(byte)0xc9 ,  (byte)0x53 ,  (byte)0x67 ,
(byte)0x9a,  (byte)0x5b,  (byte)0xc8,  (byte)0xae,  (byte)0x18 };
```

以上就是 NetworkDispatcher 类的声明和相关变量的定义，要完成整个类的功能，还需要完成数据的接收方法、加密方法、发送方法等，下面就分别讲解这几个方法的实现过程。

数据加密的过程在 NetworkDispatcher 类中由 SetKey()方法来完成，接收一个字符串类型的明文作为参数，通过加密机制对此明文进行加密。实现如下：

```
   /**
    *SetKey()方法功能主要用来对明文数据进行加密
     * return_type : void
     * @param iKey，表示字符串类型的明文信息
     */
    //定义一个 SetKey()方法，完成加密功能，传入一个字符串类型值作为参数
    public void SetKey(String iKey){             //设置 key 值
        if(iKey==null || iKey.length()<=0){      //对 key 值进行判断
            //如果参数值为 null 或不存在，则设置 keySet 标记为 false
            keySet=false;
            return;
        }
//以下就是在 Java 中，对数据加密的实现过程，关于 Java 中数据加密的机制和方法，在上文中
已经有所讲解。下面的这段代码就是 Java 加密编程方法的具体实现，读者需结合上文的讲解来
理解
        keySet=true;                     //如果 kyeSet 标记为 true，执行以下加密过程
   try {
       Provider sunJce = new SunJCE();        //新建一个 SunJCE 对象
       Security.addProvider(sunJce);          //将 SunJCE 的实例传给 Security
       paramSpec = new PBEParameterSpec(salt, 20);  //调用 PBE 加密方法
       //输入参数转为字符数组，传给 PBEKeySpec 构造方法，产生一个 PBEKeySpec 对象
       PBEKeySpec keySpec = new PBEKeySpec (iKey.toCharArray () );
       //使用 PBEWithMD5AndDES 加密算法为数据加密
       SecretKeyFactory keyFactory = SecretKeyFactory.getInstance
       ("PBEWithMD5AndDES");
       //根据提供的密钥规范：keySpec、keyFactory 生成私钥对象
       secretKey = keyFactory.generateSecret(keySpec);
       //调用 Cipher 的 init()方法，进行初始化，完成相应的加密、解密功能
```

```
        encCipher.init(Cipher.ENCRYPT_MODE, secretKey,  paramSpec);
        decCipher.init(Cipher.DECRYPT_MODE, secretKey,  paramSpec);
      //捕获并处理相应异常
      } catch (Exception ex) {
         ex.printStackTrace ();
      }
   }
```

以上的方法就完成了数据的加密。但需要注意的是，本文中的代码只作重点说明，并不完整，要完成全部的加密解密流程，还需要其他的辅助方法，具体的实现请参考源代码。

完成了加密以后，NetworkDispatcher 类还需要完成数据的分发功能，数据分发是向所有的结点发送数据消息，用一个名为 DispatchToAll 的方法表示，根据发送数据的不同，此方法有 3 个重写的方法，具体实现如下：

```
   /**
   *DispatchToAll()方法，用于虚拟 P2P 网络进行数据的分布，将消息从一个对等点通过多播
   *的方式分发给其他所有的对等点，根据传入的参数不同，它有 3 个重写的方法，具体实现如下：
   */

//以下方法是 abstract 类型的 DispatchToAll()方法，传入的参数为：(byte []iBuf,int
iSize)，因为它是 abstract 类型的，所以只有方法声明，没有具体实现的方法体，具体由
MulticastDispatcher 类来继承并实现此方法，关于 MulticastDispatcher 类，后文会讲到
 protected abstract void DispatchToAll(byte []iBuf,int iSize) throws
 Exception;

//以下是 DispatchToAll 的第二个重写的方法，传入的参数为：(byte []iBuf)，在方法体的
实现中，调用 DispatchToAll(iBuf,iBuf.length)来执行，主要是将一个 byte 数组的数据
内容分发出去
    protected void DispatchToAll(byte []iBuf) throws Exception{
        DispatchToAll(iBuf,iBuf.length);
    }

//以下是 DispatchToAll 的第三个重写的方法，传入的参数为(Message iMsg)，Message 类
在后文会讲到，是一个已经封装好的消息类型，方法体的实现过程如下：
    public void DispatchToAll(Message iMsg){
        try {
            //新建一个名为 bOut 的字节数组输出流
            ByteArrayOutputStream bOut= new ByteArrayOutputStream ();
            //用一个 Boolean 类型的 EncData 标识数据是否需要加密
            boolean EncData=(keySet && !iMsg.DontEncrypt());
            //根据数据信息对加密要求的不同，执行不同的写入操作
            if(EncData)
                //如果数据需要加密，则在 write()方法中传入 BLOCK_ENCRYPTED 参数
                bOut.write (BLOCK ENCRYPTED);
            else
                //如果数据不需要加密，则在 write()方法中传入 BLOCK_UNENCRYPTED 参数
                bOut.write (BLOCK_UNENCRYPTED);
            OutputStream underlayingStream=bOut;
            if(EncData)
            //对当前的数据流，调用 CipherOutputStream 对其进行加密
            UnderlayingStream=new CipherOutputStream(bOut,encCipher);
            //将加密后的数据封装成 ObjectOutputStream
            ObjectOutputStream ooStream=new ObjectOutputStream (under-
            layingStream);
```

```
            //writeObject()方法，将传入的 Message 写出
            ooStream.writeObject (iMsg);
            //关闭输出流
            ooStream.close ();
            //将 bOut 对象转换成字节数组，调用 DispatchToAll()方法将其分发出去
            DispatchToAll (bOut.toByteArray ());
        //捕获并处理异常
        } catch (Exception ex) {
            ex.printStackTrace ();
        }
    }
```

以上就是 NetworkDispatcher 类中数据分发的方法。下面再讲一下数据的接收方法，数据接收的方法由 DataReceived()方法来完成，需要传入一个字节数组和一个 int 型的数据长度作为参数，以下就是 DataReceived 方法的具体实现过程。

```
 /**
  *DataReceived()方法，用来接收数据，在接收过程中，首先判断数据是否加密的，如果加密
  *则先执行一个解密过程，再将数据接收，如果是非加密的，则直接接收数据。
  */
//声明一个返回值为 void 的 DataReceived()方法，传入的参数为(byte []iBuf,int iLen)
protected void DataReceived(byte []iBuf,int iLen){
      try {
         //接收加密数据时的处理，对 iBuf 中第 0 字节的值进行判断，是否加密的
          if(iBuf[0]==BLOCK ENCRYPTED){
              if(!keySet)
                  return;
               //对数据执行解密
               byte []outBuf=new byte[decCipher.getOutputSize(iLen-1)+1];
               iLen=decCipher.doFinal(iBuf,1,iLen-1,outBuf,1)+1;
               iBuf=outBuf;
           //如果数据没有加密，则直接返回
           } else if (iBuf [0]! =BLOCK UNENCRYPTED)
              return;
          //初始化一个名为 bIn 的字节数组输入流，指定空间、起始位、长度等信息
          ByteArrayInputStream bIn=new ByteArrayInputStream (iBuf, 1, iLen-1);
          //传入 bIn 数据流，封装一个 ObjectInputStream 对象
          ObjectInputStream ooStream=new ObjectInputStream (bIn);
          //通过 readObject()方法，读取内容
          Object msgIn=ooStream.readObject ();
          //关闭对象输入流
          ooStream.close ();
          //强制转换成 Message 类型
          Message recMsg=(Message)msgIn;
          //通过 Manager 类的实例的 ParseMessage 处理此 Message 对象
          Manager.GetInstance ().ParseMessage (recMsg);
      }
      //捕获并处理异常
      catch(BadPaddingException ex){
          try {
              decCipher.init(Cipher.DECRYPT MODE,secretKey, paramSpec);
          } catch (Exception exc) {}
      }
      catch (Exception ex){
          ex.printStackTrace ();
      }
 }
```

在以上 NetworkDispatcher 类的实现过程中，用到了一个名为 salt 的静态的字节数据常量，程序中的代码是：

```
private static byte[] salt ={(byte)0xc9, …… (byte)0x18 };
```

需要说明一点的是，通常，salt 指的是一种加密算法，所谓 SALT 加密是指，加密前要在加密的数据中添加一个随机串，这样，即使是相同的数据值，加密后的结果也不同，增加了数据的安全性。这里的 salt 就是一个随机串的意思。

2．MulticastDispatcher类的实现

上文已经提到过，MulticastDispatcher 类是对 NetworkDispatcher 的继承和具体实现，它的核心是一个名为 RecvThread 的内部类，此类是一个多线程的类。另外它还实现了 NetworkDispatcher 类中的 abstract 类型 DispatchToAll()方法。下面主要就对这两个方法进行说明。

MulticastDispatcher 类在具体实现过程中，用 Java 的多播套接字机制来具体地实现数据的发送和接收。此类源代码位于 p2pchat 工程中的 p2p.chat.net 包下，详细的源代码请参考随书所附光盘。

【源代码示例：ch12\ch12_code\p2pchat\src\p2p\chat\net\MulticastDispatcher.java】

```
    /**
    * 此类主要用 IP 多播技术来发送数据，关于此技术的详细讲解，请参考本章第 3 节所讲的 Java
    * 网络编程、P2P 实现机制等方面的知识
    */
package p2p.chat.net;
import java.io.*;
import java.net.*;
//定义一个 MulticastDispatcher 类，继承 NetworkDispatcher 类
public class MulticastDispatcher extends NetworkDispatcher{
     //选用专门为多播指定的 D 类 IP 地址（224.0.0.1～239.255.255.255）任选一个即可，
       用来创建一个多播组
    public static final String DefaultMulticastGroupIP="230.1.1.1";
    //使用指定的端口（一般选 1024 以上的端口号）只要是空闲端口即可，用于建立多播套接字
    public static final int DefaultMulticastGroupPort=1314;
    //定义一个接收文件的缓冲区
    private static final int RecvBufSize=65536;
    private MulticastSocket cSock;  //定义多点 Sockets，用于 IP 多播
    private InetAddress curAddr;    //InetAddress 来获取本机名称和 IP 地址信息
    private int curPort;            //当前端口
     ///内部的私有类，用多线程的方法来执行接收过程
    private class RecvThread extends Thread{    //继承 Thread 类实现多线程
        private MulticastSocket cSock;          //定义 MulticastSocket
        private NetworkDispatcher Dispatcher;   //定义 NetworkDispatcher
        //定义接收的方法，需要传入 NetworkDispatcher 对象和 MulticastSocket 对象作
          为参数
        public RecvThread(NetworkDispatcher iDispatcher, MulticastSocket
        iSock){
            cSock=iSock;
            Dispatcher=iDispatcher;
        }
        public void run(){                      //重写 Thread 类的 run()方法
```

```
            byte inBuf[]=new byte[RecvBufSize]; //开辟缓冲区
            DatagramPacket rPack=new DatagramPacket(inBuf, RecvBufSize);
            try {
                for(;;){                            //无穷循环
                    cSock.receive(rPack);           //接收数据
                    Dispatcher.DataReceived(rPack.getData(), rPack.get-
                    Length());                      //分发
                }
            } catch (IOException ex) {              //捕获异常
                ex.printStackTrace ();
            }
        }
    }
```

在 MulticastDispatcher 类中，还实现了 NetworkDispatcher 类中的 abstract 类型的 DispatchToAll()方法，此方法主要是通过 MulticastSocket 对象的 send()方法，将数据报发送出去。具体实现如下：

```
protected void DispatchToAll(byte iBuf[],int iSize) throws Exception{
        //通过 DatagramPacket 封装数据报
        DatagramPacket dPack=new DatagramPacket (iBuf, iSize, curAddr,
        curPort);
        //调用 MulticastDispatcher 的 send()方法，发送数据报
        cSock.send(dPack);
    }
```

注意： 以上只对 MulticastSocket 类中的主要方法进行了说明，完整的实现还需要其他的辅助方法共同完成，一些编程细节也是理解此类的关键，所以读者要结合源代码来学习。

在 MulticastSocket 类的实现中，用到了 java.net.MulticastSocket 类，它是一个多播数据报套接字类，用于发送和接收 IP 多播包。MulticastSocket 是一种 (UDP)DatagramSocket，它具有加入 Internet 上其他多播主机“组”的附加功能。

多播组通过 D 类 IP 地址和标准 UDP 端口号指定。D 类 IP 地址在 224.0.0.0 和 239.255.255.255 的范围内（包括两者），其中本程序中用到的这个 D 类 IP 是 230.1.1.1。还需要注意一点，地址 224.0.0.0 被保留，不应使用。

在调用此类的时候，可以通过首先使用所需端口创建 MulticastSocket，然后调用 joinGroup(InetAddress groupAddr)方法来加入多播组即可。

12.5.3　实体类 Peer 与 Channel 的实现

在本系统中，Peer 描述的是一个个参与系统的客户端，而 Channel 则是一个模拟的 P2P 网络平台，它提供了 Peer 间通信的通道，类似于一个聊天室窗口环境。系统中可以创建多个频道，同一个频道内的 Peer 可以进行 P2P 的通信。下面就具体讲解一下这两个类的实现。

注意： 这里说 Channel 类似于聊天室，只是说它提供了一个类聊天室的环境，它与聊天室的实现原理是完全不同的。应该说本系统中的这个 Channel 与 P2P 视频系统里的 Channel 类似，在一个 P2P 视频系统中，会有多个不同的 Channels，但只有那

些登录了同一个 Channel 的 Peer 间才会共享数据信息，本系统中的 Channel 就是这个意思。

1．Peer类的实现

就 P2P 的即时通信系统的开发过程而言，Peer 必然是要首先考虑的实体对象，因为它是整个系统的核心和主要参考者，本系统中所有的功能都是围绕着 Peer 进行的。

Peer 类主要用来描述 Peer 的相关信息，包括一些特征属性和基本的方法。构造 Peer 类的时候，需要重点注意 Peer 的状态信息，因为 Peer 的状态直接影响着系统的执行动作。根据 Peer 的特征和系统对 Peer 的要求，Peer 类的类图如图 12.20 所示。

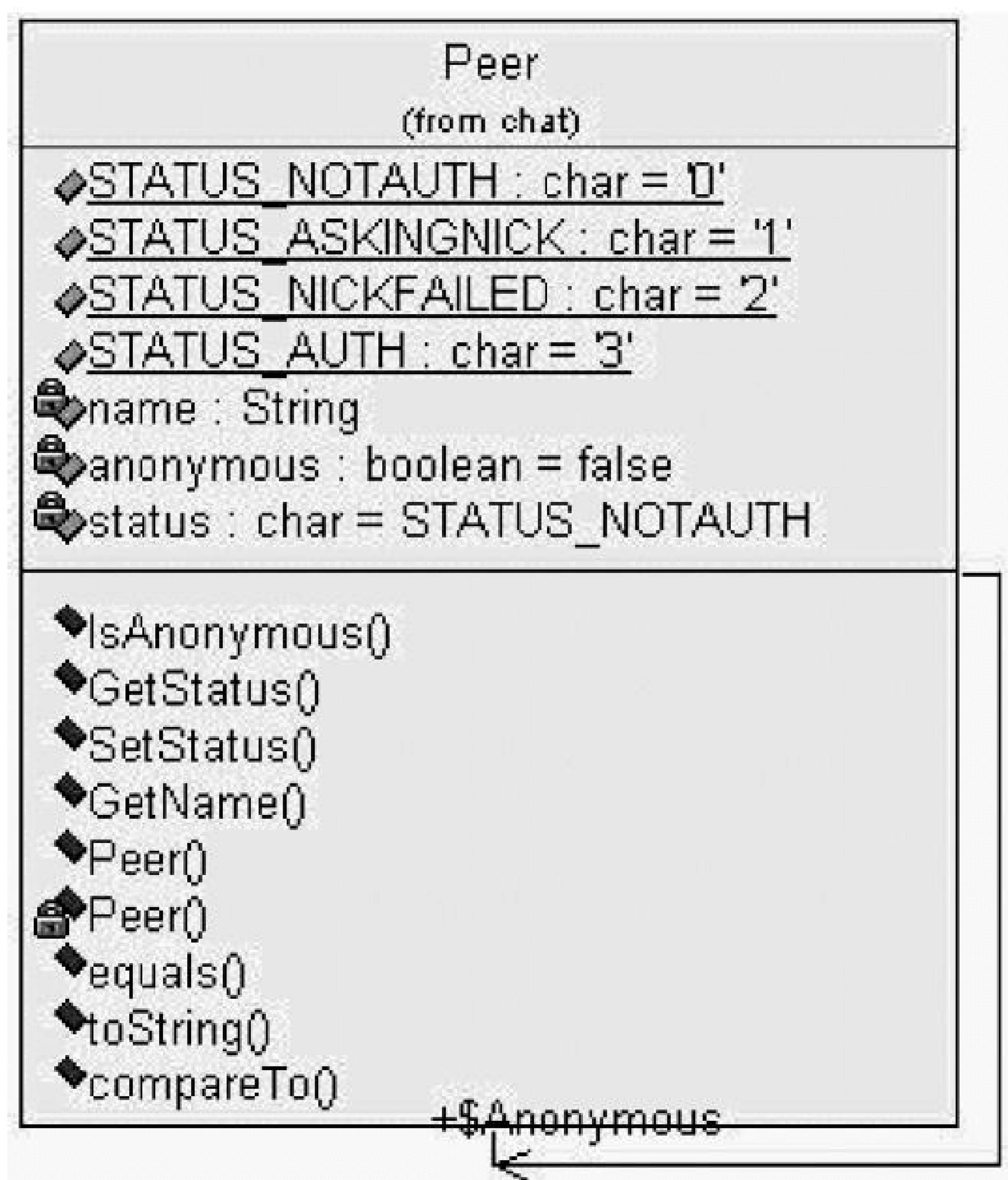


图 12.20　Peer 类类结构图

从图 12.20 中可以清楚地看到，Peer 主要有 3 个属性，分别为名称、是否匿名、状态信息。Peer 的状态分为 4 种，由 4 种不同的状态代码来表示。从方法上来看，主要有两个构造方法，一个是构造匿名的 Peer；另一个是创建一个设定名称的 Peer 实例，其他的方法还包括判断是否匿名、取得名称、设置状态、取得状态信息等。

另外，Peer 类需要实现 Serializable 和 Comparable 接口，实现 Serializable 接口，表明这个类可以直接写入一个流而进行存储，用于网络传输等操作。而实现 Comparable 接口，主要就是实现它的 compareTo()方法，该接口定义类的自然顺序。实现该接口的类就可以按这种方式排序，一般要求 e1.equals((Object)e2)和 e1.compareTo((Object)e2)==0 具有相同的值，这样的话就称自然顺序和 equals 一致，所以在此类中还重写了 equals()方法。

注意：当同一时间登录大量的 Peer 时，这就需要有一个先后的操作顺序了，所以这里有一个关于排序的相关操作，当然这需要多人同时验证才能看出效果。

Peer 实体类，主要用于描述 Peer 的相关信息，代表一个参与 P2P 聊天系统的客户端实

体。此类的源代码位于 p2pchat 工程中的 p2p.chat 包下，类的源代码请参考随书所附光盘。

【源代码示例：ch12\ch12_code\p2pchat\src\p2p\chat\Peer.java】

以下是关于 Peer 类中主要方法的说明。

```
package p2p.chat;
import java.io.Serializable;
public class Peer implements Serializable, Comparable<Object>{
     //定义一个静态的匿名 Peer
    public static Peer Anonymous=new Peer();
    //定义 Peer 的状态代码，分别用 Char 型的'0', '1', '2', '3'表示
    public static final char STATUS_NOTAUTH='0'; //当前 Peer 状态没有经过认证
    public static final char STATUS_ASKINGNICK='1';   //请求 Peer 的名称
    public static final char STATUS_NICKFAILED='2';   //Peer 的名称出现问题
    public static final char STATUS_AUTH='3';         //经过认证的 Peer 状态
    private String name;                              //Peer 的名字
    private boolean anonymous=false;                  //是否匿名
    private char status=STATUS_NOTAUTH;     //默认的 Peer 状态是没有经过认证的
   //判断此 Peer 是否匿名
    public boolean IsAnonymous(){
        return anonymous;
    }
    //取得 Peer 的状态信息
    public char GetStatus(){
        return status;
    }
    //设置 Peer 的状态
    public void SetStatus(char iStatus){
        status=iStatus;
    }
   //取 Peer 的名字
    public String GetName(){
        return name;
    }
    /** 创建一个新的 Peer 实例 */
    public Peer(String iName) {
        name=iName;
    }
    //创建一个匿名的 Peer 用户
    private Peer(){
        anonymous=true;
        name="???";
    }
 }
```

Peer 类在设计的时候，用到了创建匿名的 Peer()方法，但在本系统具体实现过程中基于简洁的要求，是不允许匿名 Peer 登录的，所以并没有用到这个功能。作为一个完善的 P2P 即时通信系统这个功能是需考虑在内的。这里就交给读者去完成，有兴趣的读者可以自己去实现 Peer 的匿名登录功能，以进一步扩展系统的应用。

2．Channel类的实现

Channel 类用来描述一个 P2P 频道的信息，当有 Peer 加入到网络中时，可以选择一个已经有的频道加入，也可以创建一个新的频道。同一个频道的所有 Peer 之间可以完成 P2P 的即时通信，包括文件共享和信息交互等。

Channel 类中除了描述频道的名称、所有者、密码等基本信息外，还要提供 Peer 加入和离开的方法、增减结点的方法、信息通知的方法等。Channel 类的设计结构图如图 12.21 所示。

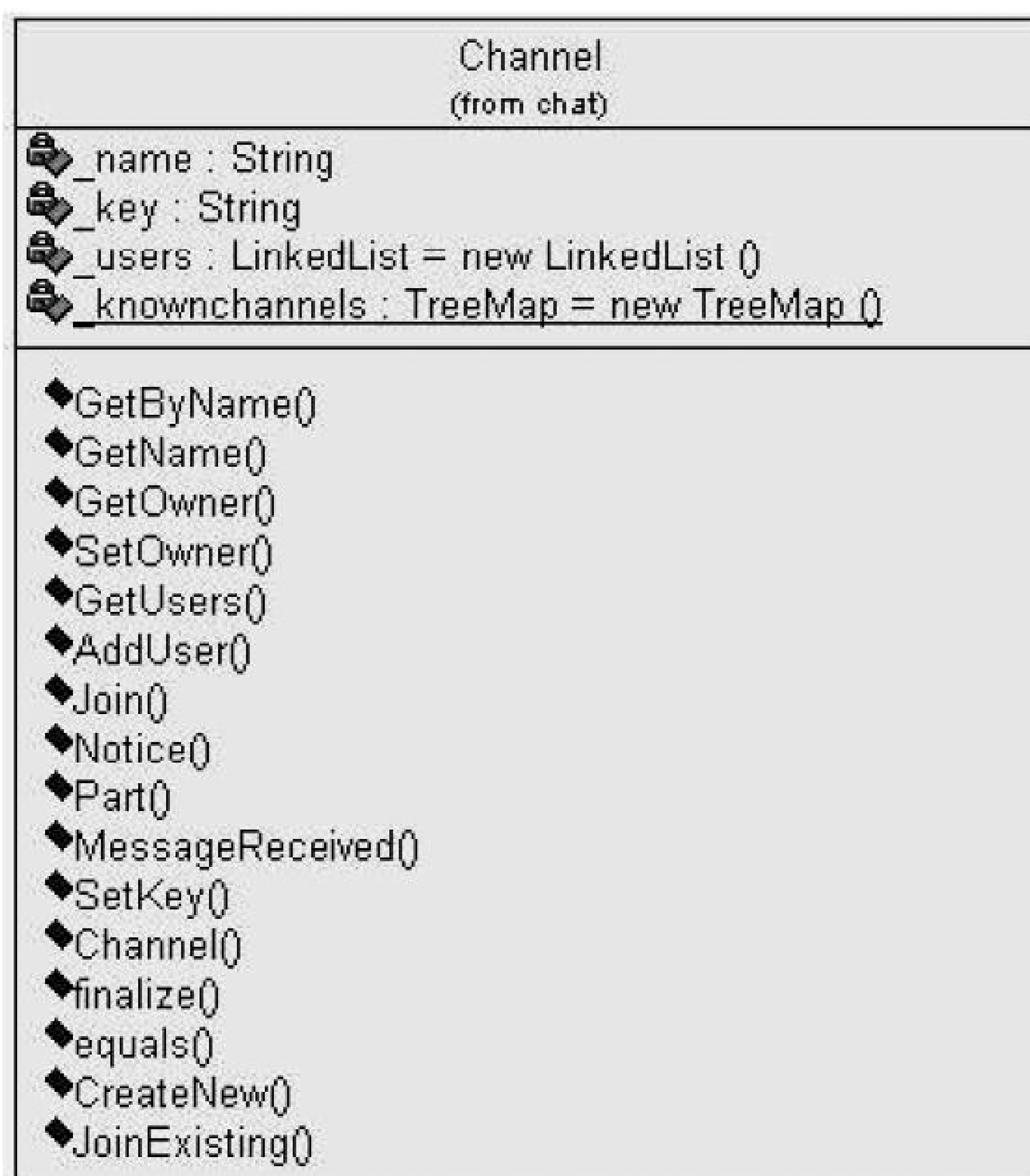


图 12.21　Channel 类的设计图

由图 12.21 可以看出 Channel 类的基本属性和方法体，在属性里主要有频道的名称、进行频道的密码、频道中所有的 Peer 及当前已存在的频道列表等。其中的一些方法主要都是针对频道信息的一些操作，这在源代码中会有说明。

Channel 实体类，主要用于描述 P2P 频道的相关信息，模拟一个 P2P 网络平台。此类源代码位于 p2pchat 工程中的 p2p.chat 包下，类的源代码请参考随书光盘，Channel 类的源代码请参考随书所附光盘。

【源代码示例：ch12\ch12_code\p2pchat\src\p2p\chat\Channel.java】

以下是 Channel 类主要方法的说明，完整的程序请参考源代码。

```
package p2p.chat;
import java.util.*;
import p2p.chat.message.*;
import p2p.chat.ui.mainFrame;
public class Channel{
    private String name;                //定义频道的名称
    private Peer owner;                 //频道的所有者，是哪个 Peer 创建的
    private mainFrame ui;               //主频道界面
    //主频道界面中的 Peer 列表
    private LinkedList<Peer> peers=new LinkedList<Peer>();
    //当前网络中已经存在的频道，用 Map 存储，一个频道名称对应一个频道实体
    private static TreeMap<String, Channel> knownchannels=new TreeMap
    <String, Channel>();
    //通过频道的名称返回一个频道实体
    public static Channel GetByName(String iName){
        if(knownchannels.containsKey(iName))
```

```
            return (Channel)knownchannels.get(iName);
        else
            return null;
    }
    public String GetName(){          //get 方法，取得名字
        return name;
    }
    public Peer GetOwner(){           //get 方法，此频道的创建者
        return owner;
    }
    //用 Peer[]型表示当前频道中所有的 Peer，也就是取得所有的用户
    public Peer[] GetPeers(){
        Peer[] outArr=new Peer[peers.size()];//将当前频道中所有的 peer 都取出来
        peers.toArray(outArr);        //转换成数组结构
        return outArr;
    }
    //添加一个新用户，也就是新加入一个 Peer 的意思
    public void AddPeer(Peer iPeer){     //在当前频道中添加一个新 Peer
        if(peers.contains(iPeer))        //判断在当前频道下是否已经存在此 peer
           return;                       //如果存在此 peer，则直接返回
        peers.add(iPeer);                //否则在 peer 列表中加入此新加入的 peer
        ui.UpdateNickList(GetPeers());   //UI 界面中更新当前 peer 列表
    }
    //加入频道
    public void Join(Peer iPeer){        //针对一个 peer 加入频道的情况
        //当有一个 peer 加入频道时，首先会有一个广播通知，告诉其他所有 peer 有新成员加入
       Notice("Peer::" + iPeer.GetName() + " 加入了频道 " + GetName());
        AddPeer(iPeer);                  //将此 peer 加入到当前 peer 列表中
    }
   //频道的通知消息
    public void Notice(String iStr){
        ui.AddRecvLine("::: "+iStr);     //通过 UI 界面来进行系统消息的发布和通知
    }
    //Peer 离开频道
    public void Part(Peer iPeer){        //当有 peer 离开频道时，就是将此 peer 从
                                          当前列表中除去
        if(peers.contains(iPeer))        //当前列表中是否包含此 peer
            peers.remove(iPeer);         //从此列表中将此 peer 移除
        //当有 peer 离开频道时，也会发出系统通知，告诉其他所有 peer，有成员离开了
        Notice("Peer::" + iPeer.GetName() + "离开了频道" + GetName());
        ui.UpdateNickList(GetPeers()); //更新 UI 界面的 peer 列表。
    }
    //由 Peer 向频道发送消息
    public void MessageReceived(ChannelMessage iMsg){
    //在 UI 界面中，peer 结点接收到系统消息的形式
    ui.AddRecvLine("< "+iMsg.GetSender().GetName() + ">说: "+ iMsg. GetText());
    }
    //设置频道密码
    public void SetKey(String iKey){
        //通过 Manager 实例调用 NetworkDispatcher 的 SetKey()方法，对密码进行加密
        Manager.GetInstance().GetDispatcher().SetKey(iKey);
    }
    /** 构造方法，创建一个新的频道实例 */
    public Channel(String iName) {
        name=iName;                          //对频道的名称赋值
        ui=new mainFrame();                  //处理 UI
```

```
        ui.setTitle("当前频道： " + name);    //设置频道的名称
        ui.setVisible(true);                  //设置频道的 UI 界面可见
        knownchannels.put(name, this);        //将新建的频道放入到已有的频道列表中
    }
    //频道退出以后，从已知频道列表中移除
    public void finalize() throws Throwable{
        knownchannels.remove(this);       //从已有的频道列表中将其移除
        super.finalize();
    }
    //执行创建频道的操作，通过名称和密码来唯一地确定一个频道
    public static void CreateNew(String iName, String iKey){
        Channel newChan=new Channel(iName);
      if(iKey!=null && iKey.length()>0)
        newChan.SetKey (iKey);
      //设置频道的创建者
      newChan.SetOwner (Manager.GetInstance().GetMe());
      //频道创建成功之后，将频道消息交给“频道广告线程”，将新建的频道信息不停地广告出去
      Manager.GetInstance().SetAndAdvertiseChannel(newChan);
    }
    //执行加入一个已有的频道操作，通过选择频道名称和输入频道的认证密码来加入此频道
    public static void JoinExisting(String iName, String iKey) throws
    Exception{
        //对用户输入的密码进行加密，然后进行密文校验
        Manager.GetInstance ().GetDispatcher().SetKey(iKey);
        //系统发出通知消息，消息内容是当前哪一个 Peer 加入了哪一个 Channel
        ServiceMessage newMsg=new ServiceMessage(Manager.GetInstance().
                                GetMe(), ServiceMessage.CODE JOIN, iName);
        //将以上的消息内容通过 IP 多点广播的形式通知给每一个 Peer
        Manager.GetInstance ().GetDispatcher().DispatchToAll(newMsg);
    }
}
```

以上就是 Channel 类的源代码，看完 Channel 类的源代码，还需要补充几点关于此类的说明。

- Peer 可以自由创建频道，或加入已有的频道。
- 频道是依附于 Peer 的存在而存在的，当频道中没有 Peer 时，频道本身也就不存在了。
- 频道不归创建者所有，而是所有的 Peer 所共有。
- 创建频道的 Peer 离开退出频道以后，只要此频道中还有其他 Peer，那么此频道就存在。
- 频道与频道之间是两个独立的数据通道。

12.5.4　消息模块的实现

本系统中，所有用于在网络中发送的东西都统一地用消息来表示，根据功能的不同、作用对象的不同，消息可有多种。本节就讲一下，本系统中所有“消息”实现的方法。

1. 消息的基本要点

从面向对象的角度来分析，作为一个消息，它要在本系统的网络传输，它应该满足这样几个条件：消息是否需要加密、此消息是由谁发送的、消息的内容是什么，因而，对每

个消息而言，它都要有 3 个基本的要点。

（1）消息的加密：需要在 Peer 间进行交互的消息都是要进行加密的，而有些消息则不需要加密。如本系统中用到的用于通信和管理的消息，此消息传送的并不是私人的数据信息，而且是必须要被接收到的。如当前存在的频道列表消息，这是不需要加密的。

（2）消息的所有者：每一个消息都必须要有一个确定的所有者，也就是说这个消息由谁发出必须是确定的。

（3）消息的内容：消息的内容指的是在 Peer 之间或 Peer 与频道之间所传送的消息实体。内容可以为空，如果不为空，其内容必须是基于文本格式的数据。

以上是每个消息都必须具备的 3 个要点，不同的消息根据其应用功能的不同还需要定义相关属性、实现其他的方法。

2．类设计图

上文已经说过，本系统中用到的消息有 3 个共同的基本要点，将它们共同的东西抽取出来构建一个父类，其他的类作为子类来继承这个父类，这样可以有效地组织代码，也是面向对象设计的重要体现。

在类的设计里已经说明了这些消息是如何继承的，这里再详细地展示一下它们的类图，此图也包括详细的属性和方法以及多个消息类之间的继承关系，如图 12.22 所示。

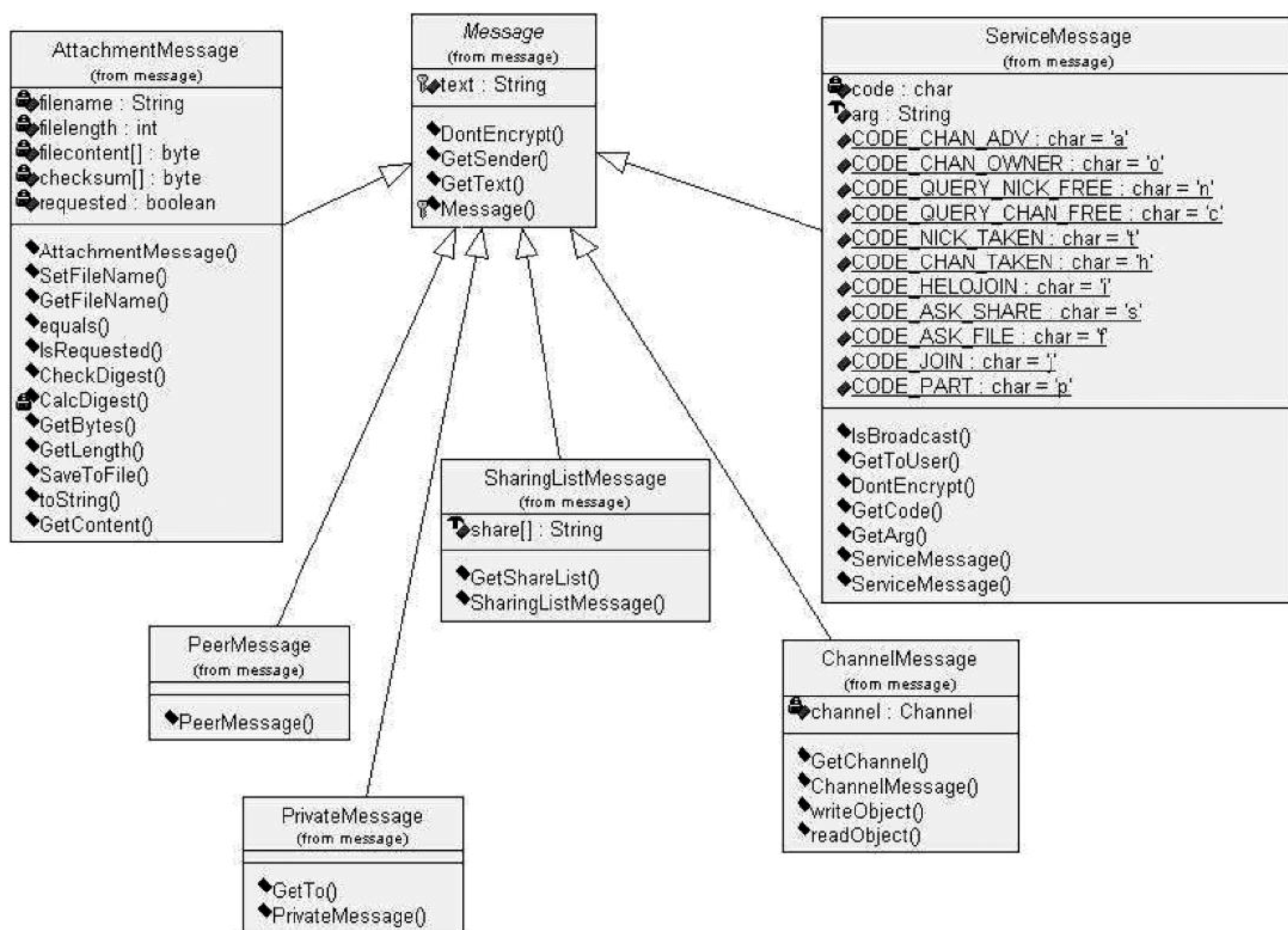


图 12.22　消息模块中所有类的类图及类之间的继承关系

图 12.22 清晰地展示了这几个消息类的关系及它们的属性和方法。根据类的名称、属性名及方法名等，可以很清楚地理解这些类的基本功能。

3．Message类的实现

Message 类是一个抽象的父类，为了进行 I/O 操作，它实现了 Serializable 接口，从类的方法体上看，它的主要方法就是围绕着消息的 3 个基本要点来实现的，下面就对 Message 类涉及的主要方法进行讲解。

Message 类，是一个抽象类，它是本系统中所有“消息类”的父类，此类的源代码位于 p2pchat 工程的 p2p.chat.message 包下。类的源代码请参考随书光盘，Message 类的代码位置如下：

【源代码：\源代码\ch12\ch12_code\p2pchat\src\p2p\chat\message\Message.java】

以下是 Message 类中主要方法的说明：

```
package p2p.chat.message;
import java.io.*;
import p2p.chat.Peer;
/**
 * @author gl
 * 抽象的虚拟类，实现了 Serializable 接口
 */
public abstract class Message implements Serializable{
    protected String text;
    protected Peer from;
    /*
     * 以下方法，主要用于实现消息的 3 个基本要点
     */
    public boolean DontEncrypt(){                        //此消息是否需要加密
        return false;
    }
    public Peer GetSender(){                             //此消息由谁发送
        return from;
    }
    public String GetText(){                             //消息的内容是什么
        return text;
    }
    protected Message(String iMsg, Peer iFrom) {     //构造方法，构造一条消息
        text=iMsg;
        from=iFrom;
    }
}
```

4．PeerMessage类的实现

从 PeerMessage 的类图中可以看出，此类只有一个 PeerMessage()的构造方法，实现起来很简单。

PeerMessage 类，继承了 Message 类，它主要描述了与 Peer 有关的消息。此类的源代码位于 p2pchat 工程的 p2p.chat.message 包下，请参考随书光盘源码。PeerMessage 类的源代码位置如下：

【源代码：\源代码\ch12\ch12_code\p2pchat\src\p2p\chat\message\PeerMessage.java】

PeerMessage 类中，只有一个主要的构造方法，如下：

```
package p2p.chat.message;
import p2p.chat.Peer;
```

```
//声明一个 PeerMessage 类，继承 Message 类
public class PeerMessage extends Message{
      //定义一个 PeerMessage 对象实例，名为 fromuser，意思为此 Message 来自哪里
      private Peer fromuser;
       //public 类型的 PeerMessage 构造方法，需要传入消息内容、发送消息的 Peer 为参数
       public PeerMessage(String iMsg, Peer iFrom) {
       //调用父类的构造方法
       super(iMsg, iFrom);
       fromuser=iFrom;
    }
   }
```

5．ChannelMessage类的实现

ChannelMessage 类中包括一个 Channel 类型的属性信息和 4 个基本方法，分别为 GetChannel()，取得频道信息；ChannelMessage()，构造方法；writeObject()，输出对象数据；readObject()，读取对象数据等。

ChannelMessage 类，同样继承了 Message 类，它主要表示的是所有的由用户发向频道的文本信息。此类的源代码位于 p2pchat 工程的 p2p.chat.message 包下，类的源代码如下请参考随书光盘，ChannelMessage 类的位置：

【源代码：\源代码\ch12\ch12_code\p2pchat\src\p2p\chat\message\ChannelMessage.java】

```
package p2p.chat.message;
import java.io.*;
import p2p.chat.*;
/**
 * @author gl
 * 继承 Message 类，并实现了 Serializable 接口
 */
//定义一个名为 ChannelMessage 的类，继承 Message，实现 Serializable 接口
public class ChannelMessage extends Message implements Serializable{
   //定义一个 private 的 Channel 类型
    private Channel channel;
    //getChannel()方法，取得当前的频道
    public Channel getChannel(){
        return channel;
    }
    //构造方法，创建一个 ChannelMessage 实例
    public ChannelMessage(String iMsg, Peer iFrom, Channel iToChan) {
        //继承父类的构造方法
        super(iMsg, iFrom);
        channel=iToChan;
    }
   //writeObject()方法，传入对象输出流，通过 writeUTF()方法，将当前频道名称输出
   private void writeObject(java.io.ObjectOutputStream out) throws
   IOException{
     out.writeUTF(channel.GetName());
   }
   //readObject()方法，传入对象输入流，通过 readUTF()方法，读取当前的频道信息
   private void readObject(java.io.ObjectInputStream in)
                                throws IOException, ClassNotFoundException{
      channel=Channel.GetByName(in.readUTF());
   }
}
```

🔔**注意：**在理解 PeerMessage.java 和 ChannelMessage.java 这两个类时，要和 Peer.java 类及 Channel.java 类结合起来理解，因为它们之间是紧密联系的。

6．PrivateMessage类的实现

PrivateMessage 类中有一个 Peer 类型的属性信息，还有一个构造方法和一个 GetTo()方法。其中 GetTo()方法表示的是当前消息是发向谁的，而构造方法主要完成消息基本要点的定义。

PrivateMessage 类，继承了 Message 类同时实现了 Serializable 接口，因为此消息是需要在 Peer 结点中进行传送的，所以必须要序列化，它主要表示的是两个 Peer 之间交互的文本消息。此类的源代码位于 p2pchat 工程的 p2p.chat.message 包下，类的源代码请参考随书光盘，PrivateMessage 类的位置如下：

【源代码:\源代码\ch12\ch12_code\p2pchat\src\p2p\chat\message\PrivateMessage.java】

以下是关于 PrivateMessage 类中主要方法的说明：

```
package p2p.chat.message;
import java.io.*;
import p2p.chat.Peer;
/**
 * @author gl
 * 继承 Message 类，实现 Serializable 接口，用来完成 peer 间的私有消息交互的功能
 */
//定义一个 PrivateMessage 类，继承 Message 类，实现 Serializable 接口
public class PrivateMessage extends Message implements Serializable{
    //声明一个 Private 的 Peer 类型，指消息到达的目的 peer
    private Peer to;
    //GetTo()方法，返回 Peer 类型，指此消息要到达的那个 Peer
    public Peer GetTo(){
        return to;
    }
    //PrivateMessage类的私有构造方法,需要传入String类型的Message和要发送的Peer
      对象
    public PrivateMessage(String iMsg, Peer iFrom, Peer iTo) {
        super(iMsg, iFrom);
        to=iTo;
    }
}
```

7．SharingListMessage类的实现

SharingListMessage 类继承自 Message 父类，所以满足消息的 3 个基本要点，它有一个 String 类型的 shar[]数组，表示共享的文件列表，一个 GetShareList()方法，用来取得共享列表，有一个构造方法，用来得到共享列表的内容。

SharingListMessage 类所描述的是共享文件列表的一些信息，此类的源代码位于 p2pchat 工程的 p2p.chat.message 包下，源代码请参考随书光盘，SharingListMessage 类的位置如下：

【源代码：\ch12\ch12_code\p2pchat\src\p2p\chat\message\SharingListMessage.java】

```
package p2p.chat.message;
```

```
import java.io.*;
import p2p.chat.Peer;
/**
 * @author gl
 * 继承 Message 类，实现 Serializable 接口，用来描述共享文件列表的一些信息
 */
//定义一个 SharingListMessage 类
public class SharingListMessage extends Message implements Serializable{
    //定义一个 String 数组，用来存储共享列表信息
    String[] share;
    // GetShareList()方法，取得共享列表，返回一个 String 数组
    public String[] GetShareList(){
        return share;
   }
   //通过 SharingListMessage 类的构造方法创建一个 SharingListMessage 消息实例
   public SharingListMessage(Peer iFrom, String[] iSharingList) {
       //声明父类的构造方法
        super("", iFrom);
        share=iSharingList;
    }
}
```

8. AttachmentMessage类的实现

AttachmentMessage 类，主要用来表示由 Peer 结点向网络中发送文件的消息，比如文件的名称、大小、内容等都由此类来表示。

AttachmentMessage 类的基本属性如下。

- String filename：指文件名。
- Int filelength：文件大小。
- Byte[] filecontent：文件内容。
- Byte[] checksum：文件校验值，用来检查文件的完整性，本类中用 MD5 算法实现。
- Boolean requested：用来表证这个文件是请求其他 Peer 结点的共享文件列表得到的，还是由 Peer 结点主动发送得到。

AttachmentMessage 类的主要方法如下：（这里只讲解几个主要的方法，其他的方法请参考源代码的说明）。

- AttachmentMessage()：构造方法，描述一个完整的文件信息。
- SetFileName()：设置文件名。
- GetFileName()：取得文件名。
- SaveToFile()：将文件存储到本地。
- GetContent()：读取文件内容。

AttachmentMessage 类，在系统的文件发送与接收、共享文件显示、内容读取等功能的实现上有重要作用。此类的源代码位于 p2pchat 工程的 p2p.chat.message 包下，源代码请参考随书光盘，AttachmentMessage 类的位置如下：

【源代码：\ch12\ch12_code\p2pchat\src\p2p\chat\message\AttachmentMessage.java】

AttachmentMessage 类的主要编程步骤及方法说明如下：

```
package p2p.chat.message;
import java.io.*;
```

```
import java.security.*;
import java.util.*;
import p2p.chat.Peer;
//定义一个 AttachmentMessage 类，继承 Message 类，并实现 Serializable 接口
public class AttachmentMessage extends Message implements Serializable {
    private String filename;                //文件名
    private int filelength;                 //文件大小
    private byte[] filecontent;             //文件内容
    private byte[] checksum;                //文件的完整性校验
    //用来表示此文件是被 Peer 发送过来的，还请查看其他 Peer 共享得到的
    private boolean requested;
    //构造方法，用于创建一个 AttachmentMessage 实例
    public AttachmentMessage(Peer iFrom, File iFile, boolean iReq) throws
    Exception {
        super(iFile.getName(), iFrom);      //调用父类的构造方法
        filename=iFile.getName();           //取文件名
        filelength=(int)iFile.length();     //取文件大小
        filecontent=new byte[filelength];   //取文件的内容
        requested=iReq;                     //此文件的请求者
        FileInputStream fIn=new FileInputStream(iFile);     //文件读写通道
        fIn.read(filecontent);                              //读文件内容
        fIn.close();
        checksum=CalcDigest();
    }
    public void SetFileName(String iName){ //设置文件的名称
        filename=iName;
    }
    public String GetFileName(){            //取得文件的名称
        return filename;
    }
   //重写 equals()方法，判断两个文件的比较情况
   public boolean equals (Object obj){
        //如果判定的两个文件不相等，则返回 false
        if(!(obj instanceof AttachmentMessage))
            return false;
        //将传入的对象强制转换成 AttachmentMessage 类型
        AttachmentMessage tMsg=(AttachmentMessage)obj;
     //对文件名和文件内容同时比较，只有二者都相等时，才认识两个文件相等
     return GetSender().equals(tMsg.GetSender()) && filename.equals
    (tMsg.filename);
    }
    //通过校验值检查文件的完整性
    public boolean CheckDigest() throws Exception{
        return Arrays.equals(checksum, CalcDigest());
   }
     //通过 MD5 算法对文件进行校验加密
     private byte[] CalcDigest() throws Exception{
        //返回实现指定摘要算法的 MessageDigest 对象，这里指定用 MD5 算法
        MessageDigest md=MessageDigest.getInstance("MD5");
        return md.digest(filecontent)
    }
    //取得文件内容，将文件内容数据存储到字节数组中
    public byte[] GetBytes(){
        return filecontent;
    }
    //取得文件的长度，用 int 型的数据表示文件的长度
    public int GetLength(){
```

```
        return filelength;
    }
    //将文件存储到本地
    public void SaveToFile(String iPath) throws Exception{
        FileOutputStream fOut=new FileOutputStream(iPath);  //新建一个输出流
        fOut.write(filecontent);                            //将文件内容输出
        fOut.close();                                       //关闭输出通道
     }
    //重写toString()方法，显示文件内容
    public String toString(){
        return "[" + GetSender() + "] " + filename + " ( " + filelength +
        "bytes)";
    }
    //显示文件内容
    public String GetContent(){
       char []chArr=new char[filecontent.length];
                                                 //设置一个存放文件内容的缓冲区
       for(int i=0;i<filecontent.length;i++)     //循环读取文件内容
          chArr[i]=(char)filecontent[i];         //将文件内容放到缓冲区中
       return new String(chArr);
    }
}
```

9. ServiceMessage类的实现

ServiceMessage 类：主要是由 P2P 聊天系统发出的用于同步和管理的消息，比如通知 Peer 结点加入或退出的消息等，就属于 ServiceMessge。在这个类中定义了如下几个静态常量：

```
public final static char CODE_CHAN_ADV='a';
public final static char CODE_CHAN_OWNER='o';
public final static char CODE_QUERY_NICK_FREE='n';
public final static char CODE_QUERY_CHAN_FREE='c';
public final static char CODE_NICK_TAKEN='t';
public final static char CODE_CHAN_TAKEN='h';
public final static char CODE_HELOJOIN='i';
public final static char CODE_ASK_SHARE='s';
public final static char CODE_ASK_FILE='f';
public final static char CODE_JOIN='j';
public final static char CODE_PART='p';
```

这些常量主要用来定义某些特定的消息代码，比如 CODE_CHAN_ADV='a '指的是“广告频道消息”的代码为 a。为什么要这样做呢？在上文已经说过，在网络中传输的有些消息是不需要加密的，就像这个代码为 CODE_CHAN_ADV 的消息，它主要是系统用来向网络中广告当前频道信息，没有加密的必要。因而将这些不需要加密的消息都标注出来，用简单的代码来表示，便于在程序中进行处理。

ServiceMessage 类中的 DontEncrypt()方法是对这些常量的具体应用，当代码为以上这些消息时，不执行数据的加密，从方法的名称也可以看出，DontEncrypt 就是不加密的意思。

ServiceMessage 类的源代码位于 p2pchat 工程的 p2p.chat.message 包下。类的源代码请参考随书光盘，ServiceMessage 的位置如下：

【源代码：\ch12\ch12_code\p2pchat\src\p2p\chat\message\ServiceMessage.java】

ServiceMessage 类中，主要的编程步骤及方法说明如下：

```
package p2p.chat.message;
import java.io.*;
import p2p.chat.Peer;
//定义一个 ServiceMessage 类，继承 Message 类，实现 Serializable 接口
public class ServiceMessage extends Message implements Serializable{
    //定义一个名为 code 的字符型变量
    private char code;
    //定义一个名为 arg 的字符串变量
    String arg;
    //定义一个名为 to 的 Peer 对象，表示消息要到达的 Peer
    Peer to;
    //给每一个不同消息定义一个不同的编号，用来区分哪些消息是不需要加密的
    public final static char CODE_CHAN_ADV='a';
    public final static char CODE_CHAN_OWNER='o';
    public final static char CODE_QUERY_NICK_FREE='n';
    public final static char CODE_QUERY_CHAN_FREE='c';
    public final static char CODE_NICK_TAKEN='t';
    public final static char CODE_CHAN_TAKEN='h';
    public final static char CODE_HELOJOIN='i';
    public final static char CODE_ASK_SHARE='s';
    public final static char CODE_ASK_FILE='f';
    public final static char CODE_JOIN='j';
    public final static char CODE_PART='p';
    //消息编号定义结束
    public boolean IsBroadcast(){        //判断此消息是否广播消息
        return to==null;
    }
    //DontEncrypt()方法，对不同的消息编码类型进行判断，用来确定哪些消息是不需要加密的
    public boolean DontEncrypt(){
    if(code==CODE_CHAN_ADV||code==CODE_QUERY_NICK_FREE||code==
            CODE_QUERY_CHAN_FREE||code==CODE_NICK_TAKEN
                                             ||code==CODE_CHAN_TAKEN)
            return true;
        return false;
}
     // GetCode()方法，取得消息的编码
    public char GetCode(){
        return code;
    }
    //GetArg()方法，取得消息内容
    public String GetArg(){
      return arg;
     }
     //public 型的 ServiceMessage 构造方法，传入两个 Peer 对象、消息编码、消息内容为
       参数
      public ServiceMessage(Peer iFrom,Peer iTo,char iCode,String iArg) {
        this(iFrom,iCode,iArg);
        to=iTo;
     }
     //ServiceMessage 类的另一构造方法，需要传入消息来自的 Peer 对象、消息编码、消息
       内容为参数
    public ServiceMessage(Peer iFrom,char iCode,String iArg) {
        super("",iFrom);
        code=iCode;
        arg=iArg;
    }
}
```

以上就是整个消息模块的实现，消息模块在系统中扮演着重要的角色，所有经由网络

进行发送的“东西”，都由消息模块的不同消息类来表示，在消息展示、数据交互、系统通信等各方面都有重要的作用。

12.5.5　基本功能模块的实现

本系统中用于实现基本功能的类主要有 3 个，分别为 ChanAdvertiserThread.java、FileSharing.java 和 PrivateConversation.java。下面就主要讲一下这 3 个基本类的设计与实现。

1．广告频道信息的功能

当网络中已经有一个频道创建好之后，这个频道就需要提供给其他的 Peer 加入，要加入这个频道，其前提是网络中的 Peer 结点必须知道这个频道的相关信息。ChanAdvertiser Thread 类就负责以多线程的方式不停地向网络中广播当前的频道信息，当对此信息感兴趣的 Peer 结点收到这个广播信息后，就会更新自身的频道列表，这样，每个新建的频道都能及时地被网络中的 Peer 结点为捕获到。

ChanAdvertiserThread 类，是一个多线程的类，它继承了 Java 线程中的 Thread 类，重写了 run()方法，以多线程的方式每隔 2 秒钟就向网络中广告一次当前的频道信息。此类的设计结构图如图 12.23 所示。

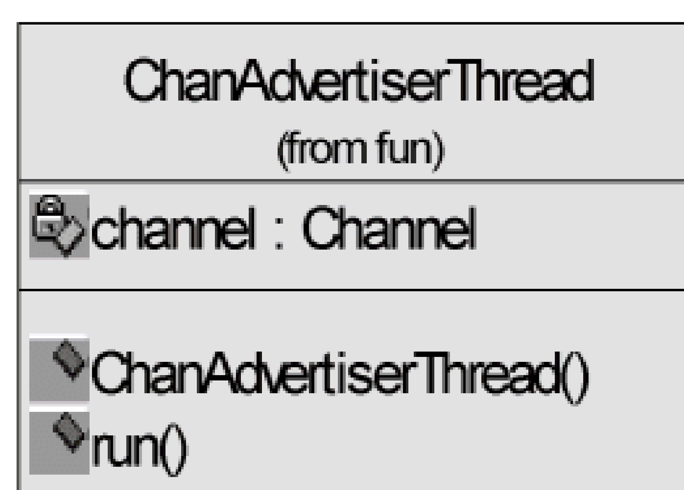


图 12.23　ChanAdvertiserThread 类的设计图

从设计类图上就可以看出这个类很简单，只有一个属性，就是此线程要广告的对象，方法上只有两个，一个构造方法，用于封装频道信息，另一个是重写的 run()方法，用于实现多线程执行。

ChanAdvertiserThread 类，继承了 Thread 类，并重写 run()方法来实现多线程的操作，用来向网络中广播当前的频道信息。此类的源代码位于 p2pchat 工程的 p2p.chat.fun 包下。类的源代码请参考随书光盘，ChanAdvertiserThread 类的位置如下：

【源代码：\ch12\ch12_code\p2pchat\src\p2p\chat\fun\ChanAdvertiserThread.java】

ChanAdvertiserThread 类中主要编程步骤及方法说明如下：

```
package p2p.chat.fun;
import p2p.chat.*;
import p2p.chat.message.*;
/**
 * @author gl
 * 多线程的方式实现，用来向网络中广告当前网络中的频道信息
 */
//定义一个 ChanAdvertiserThread 类，继承 Thread 类，用来实现多线程
public class ChanAdvertiserThread extends Thread {
    //定义一个 channel 的 Channel 变量
    private Channel channel;
```

```
    //public 类型的 ChanAdvertiserThread 类构造方法，传入一个 Channel 对象作为参数
    public ChanAdvertiserThread(Channel iChan){
        channel=iChan;
  }
    //继承 Thread 类，重写 run()方法，实现多线程方法
    public void run(){
        //调用 Manager 的 GetInstance()方法，取得一个 Manager 实例
        Manager manager=Manager.GetInstance();
        //for 循环，参数为空，说明是无限循环
        for(;;){
            try {
              //通过 ServiceMessage 对象，取得要广告的消息
                ServiceMessage advMsg=new ServiceMessage(manager.GetMe(),
                            ServiceMessage.CODE CHAN ADV,channel.GetName());
                //通过 Manager 的 DispatchToAll()方法,将广告消息向每个 Peer 广播出去
                manager.GetDispatcher().DispatchToAll(advMsg);
                //让进程休眠 2 秒钟，表示每隔 2 秒广播一次消息
                Thread.sleep(2000);
            //捕获并处理相应异常
            } catch (InterruptedException ex) {
                ex.printStackTrace();
            }
        }
    }
}
```

2. 文件共享功能

文件共享主要用来实现 Peer 间的文件共享操作，它包括两个方面的操作内容。

- 主动提供共享：Peer 结点可主动的将本地的一个目录或文件共享给其他的 Peer。
- 查看下载共享：Peer 结点可以查看本频道中所有其他结点的共享情况，也可以将其他 Peer 结点共享的文件下载到本地。

文件共享是即时通信系统的一个主要应用，也是 P2P 机制的重要体现，本系统中实现这一功能由 FileSharing 类来完成，此类的类结构图如图 12.24 所示。

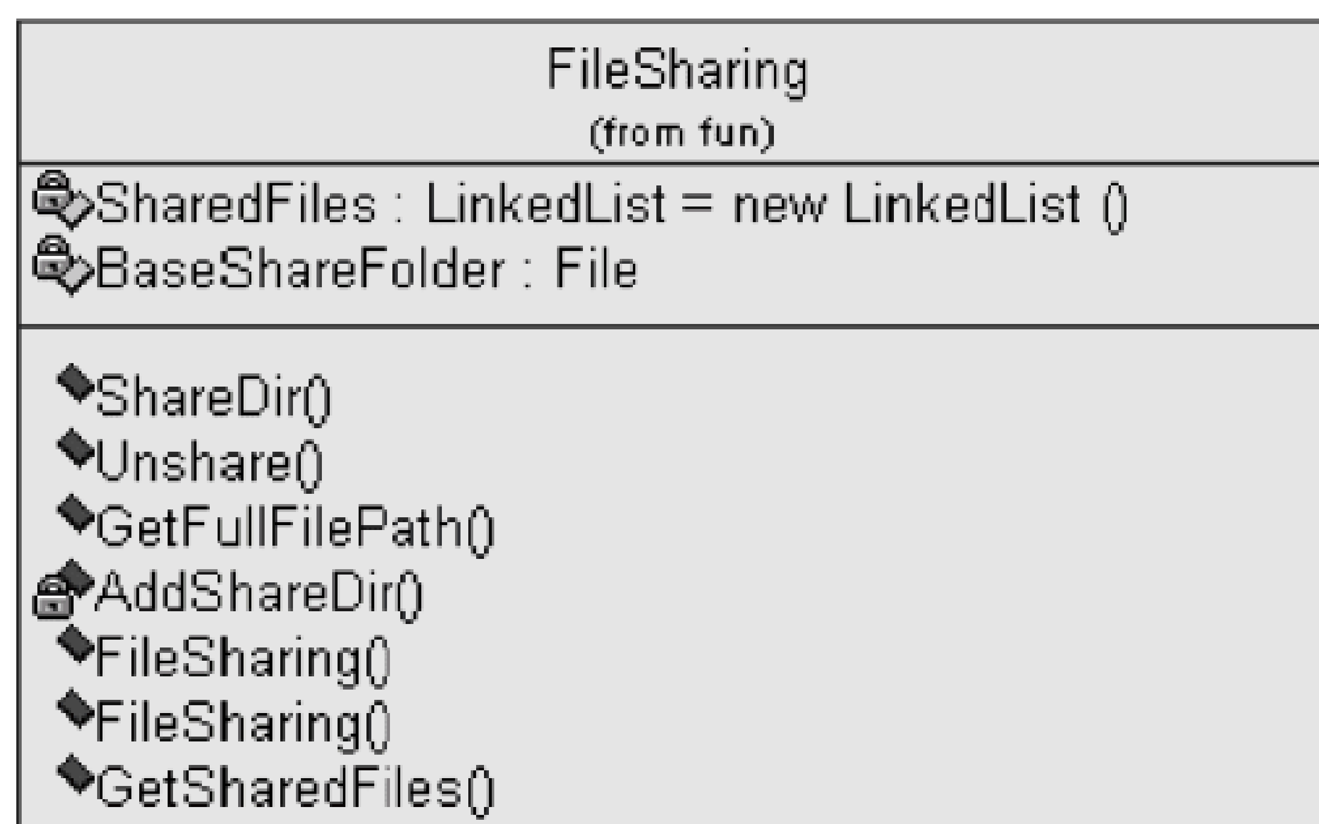


图 12.24　FileSharing 类的类图

由图 12.24 所示的类结构可以清楚地看出，FileSharing 类有两个基本的属性，一个是 LinkedList 类型的 ShareFiles，表示的是当前共享的文件列表；另一个 File 类型的 BaseSharFolder，表示的是共享的基本目录。

FileSharing 类的基本方法描述了 Peer 对共享文件的基本操作，共享目录、取消共享、得到路径、加入共享目录、取得共享文件等，具体的方法实现请参阅其源代码。

FileSharing 类，用来实现文件共享的相关功能。此类的源代码位于 p2pchat 工程的 p2p.chat.fun 包下。源代码请参考随书光盘，FileSharing 类的位置如下：

【源代码：\ch12\ch12_code\p2pchat\src\p2p\chat\fun\FileSharing.java】

以下是 FileSharingo 类中主要的编程步骤及方法说明。

```
package p2p.chat.fun;
import java.io.File;
import java.util.*;
import p2p.chat.Manager;
/**
 * @author gl
 * FileSharing 类，主要用于实现文件共享的功能
 */
public class FileSharing {
     //定义一个 LinkedList 类型的数据结构，用于存储共享的文件列表
    private LinkedList<File> SharedFiles=new LinkedList<File>();
    //定义一个名为 BaseShareFolder 的 File 类型变量，用来表示基本的共享目录
    private File BaseShareFolder;
    //ShareDir()方法，传入一个 String 类型的文件路径，用来表示共享目录
    public void ShareDir(String iPath){
        //根据文件路径，新建一个基本共享目录
        BaseShareFolder=new File(iPath);
        //将当前的文件共享列表清空
        SharedFiles.clear();
        //对当前的共享目录进行判断，判断此目录是否存在
        if(!BaseShareFolder.exists())
        //如果存在此目录，直接返回
            return;
        //否则，将此目录加入到已有的共享目录列表里，下文会讲到 AddShareDir()方法
        AddShareDir(BaseShareFolder);
    }
    //Unshare()方法，用于取消共享，只需将共享目录置为 null，共享文件列表也清空即可
    public void Unshare(){
        //将共享目录置为 null
        BaseShareFolder=null;
        //将共享文件列表清空
        SharedFiles.clear();
    }
    //GetSharedFiles()方法，用于取得共享文件的列表，返回一个 String 数组
    public String[] GetSharedFiles(){
        //初始化一个名为 outArr 的 String 数组
        String[] outArr=new String[SharedFiles.size()];
        //对共享文件列表进行遍历，取出所有的共享路径，将结果存储到 outArr 中
        for(int i=0;i<SharedFiles.size();i++){
           outArr[i]=((File)SharedFiles.get(i)).getAbsolutePath().
                                  substring(BaseShareFolder.getAbsolutePath().
                                  length());
        }
        //对 outArr 中的数据进行排序
        Arrays.sort(outArr);
        //返回 outArr 中的结果
        return outArr;
    }
    //GetFullFilePath()方法，用于读取一个共享文件的全路径信息，传入 Strign 类型文件
      路径为参数
    public String GetFullFilePath(String iFile){
```

```
        //对共享目录进行判断，是否为 null，如果是，直接返回空串
        if(BaseShareFolder==null) return "";
        //如果共享文件目录不为 null，则返回此目录的全路径
        return BaseShareFolder.getAbsolutePath()+iFile;
    }
    //AddShareDir 用于增加一个共享目录到共享目录列表中，传入 File 对象作为参数
    private void AddShareDir(File iDir){
        //通过 File 对象的 listFiles()方法，将此 File 目录的所有文件写入到 File 数组中
        File[] fList=iDir.listFiles();
        //对 File 数组进行遍历，取出所有的文件
        for(int i=0;i<fList.length;i++){
            //如果此目录下是单个文件，则在共享列表加入此文件
            if(fList[i].isFile()){
                //对文件大小进行判断，如超出指定要分发的文件大小范围，则舍弃
                if(fList[i].length()>Manager.GetInstance().GetDispatcher().
                GetMaxFileSize())
                    continue;
                //将满足条件的文件，加入到共享文件列表中
                SharedFiles.add(fList[i]);
            //如果 File 目录上是一个子目录，则递归地调用，进行递归遍历
            }else if(fList[i].isDirectory())
                //递归地调用自身，对目录进行层次遍历
                AddShareDir(fList[i]);
        }
    }
    //一个参数为空的构造方法
    public FileSharing() {
   }
    //另一个需要传入字符串型的共享目录路径的构造方法
    public FileSharing(String iShareFld) {
        ShareDir(iShareFld);
    }
}
```

3．私人会话功能

上文已经说过，当 Peer 在加入到某一频道后，此频道中的所有 Peer 都共享一个类似于聊天室的会话环境。一个 Peer 发布的消息会通过频道广播给其他所有的 Peer，类似于聊天室或是 QQ 里的群组消息。

本系统也提供点对点的私人会话功能，也就是 Peer 结点可以从 Peer 列表中选择任意一个 Peer 与之建立私人会话。此时的会话消息会在两个对等的 Peer 结点之间传输，属于私有的会话通道，私人会话消息对系统和其他 Peer 而言是不可见的。私人会话的功能由 PrivateConversation 类来实现，此类的设计图如图 12.25 所示。



图 12.25　PrivateConversation 类的类图

由图 12.25 可见，PrivateConversation 的设计也非常简单，它只有一个属性，Peer 类型的 to，也就是这个私人会话是要发向谁的。在方法中，包括一个 GetTo()方法，要向哪个 Peer 发起会话，Show()方法是自动弹出消息对话框，提醒用户有会话消息到达并显示会话内容。（这里会用到后面的界面显示相关的类），SendMessage()和 MessageArrival()方法，就用来处理消息的发送和接收，详细的方法实现过程请参阅此类的源代码。

PrivateConversation 类，用来实现两个 Peer 间的私人会话功能。此类的源代码位于 p2pchat 工程的 p2p.chat.fun 包下。源代码请参考随书光盘，PrivateConversation 类的位置如下：

【源代码：\ch12\ch12_code\p2pchat\src\p2p\chat\fun\PrivateConversation.java】

PrivateConversation 类主要的编程步骤及方法说明如下：

```
package p2p.chat.fun;
import p2p.chat.*;
import p2p.chat.ui.privateConvFrame;
/**
 * @author gl
 * PrivateConversation 类主要用于实现 Peer 间的私人会话功能
 */
//定义一个 public 类型的 PrivateConversation 类
public class PrivateConversation {
    private privateConvFrame ui;            //两 peer 间私有会话的 UI 界面
    private Peer to;                        //定义会话是由哪个 peer 发起的
    public PrivateConversation(Peer iTo) {  //私有会话的方法
        to=iTo;                             //会话的对象
        ui=new privateConvFrame(this);      //会话的 UI 平台
    }
    public Peer GetTo(){                //GetTo()方法，返回个会话到达的 Peer 对象
        return to;
    }
    public void Show(){                 //使当前的私人会话界面可见
        ui.setVisible(true);            //设置 UI 的 setVisible()方法为 true
    }
    //MessageArrival()方法，用来处理消息到达时的操作，传入一个 String 类型的消息参数
    public void MessageArrival(String iMsg){
        //调用 UI 类的 AddRecvLine()方法，用来显示一条消息到达时会话界面上的显示格式
        ui.AddRecvLine("<" + to.GetName()+">说：  " + iMsg);
        //对当前的界面进行判断，当有新消息时，自动弹出会话界面
        if(!ui.isVisible()){
            ui.setVisible(true);
            //将会话界面上的焦点，自动定位到用户发消息的输入框中
            ui.requestFocus();
        }
    }
    // SendMessage()方法，处理发送消息的操作，传入一个要发送的消息作为参数
    public void SendMessage(String iMsg){
        //取得 Manager 的一个实例，并通过 SendPrivateMsg()方法，将消息发送出去
        Manager.GetInstance().SendPrivateMsg(to, iMsg);
        //在当前 UI 会话界面上显示消息发送的形式
        ui.AddRecvLine("<" + Manager.GetInstance().GetMe().GetName() +">说：
        " + iMsg);
    }
}
```

注意：在 PrivateConversation 类，包括以上的一些类中，用到了对其他类的引用，如 Manager 类、UI 类等，这些都是本系统涉及的其他模块中的类，在后文都会讲到。

12.5.6　界面模块的实现

界面是系统运行不可缺少的元素，通过界面可以给用户提供方便的操作和统一的展示接口。界面模块是一系列实现系统界面的类的总称，包括了本系统中用到的所有与界面实现有关的类，这些类在系统开发的时候都被封装在一个叫 p2p.chat.ui 的包里。本节就重点讲解界面模块的实现。

1．登录/注册界面

当系统启动以后，首先会弹出一个登录/注册的界面，这个界面由 chatStart.java 类来实现，它所完成的就是登录或是注册的功能。

登录指的是 Peer 结点登录一个已有的频道，注册指的是 Peer 结点注册一个新的频道，根据这样的要求，那么这个界面应该提供这样几个基本的功能。

（1）提供 Peer 结点名称的输入：每个 Peer 结点在加入网络之前必须要唯一地标识自己的名字。

（2）提供频道列表的选择和认证密码的输入：如果 Peer 要加入一个已有的频道，那么就要在这个界面中列出已有的频道列表，还要提供一个密码输入框，因为每个频道需要密码才能进行输入。

（3）提供创建新频道的名称和密码的功能：如果 Peer 结点要自行创建一个新的频道，那么，就需要提供一个频道名称、频道密码的输入框，这是创建一个新频道的前提条件。

要实现以上几个基本的功能，就需要在界面中加入相应的标签、输入框、列表框、按钮等元素，chatStart 作为一个界面类，它继承了 JFrame 类，而且用单例模式来保证 chatStart 在调用时只用一个实例，关于 chatStart.java 类的设计类图如图 12.26 所示。

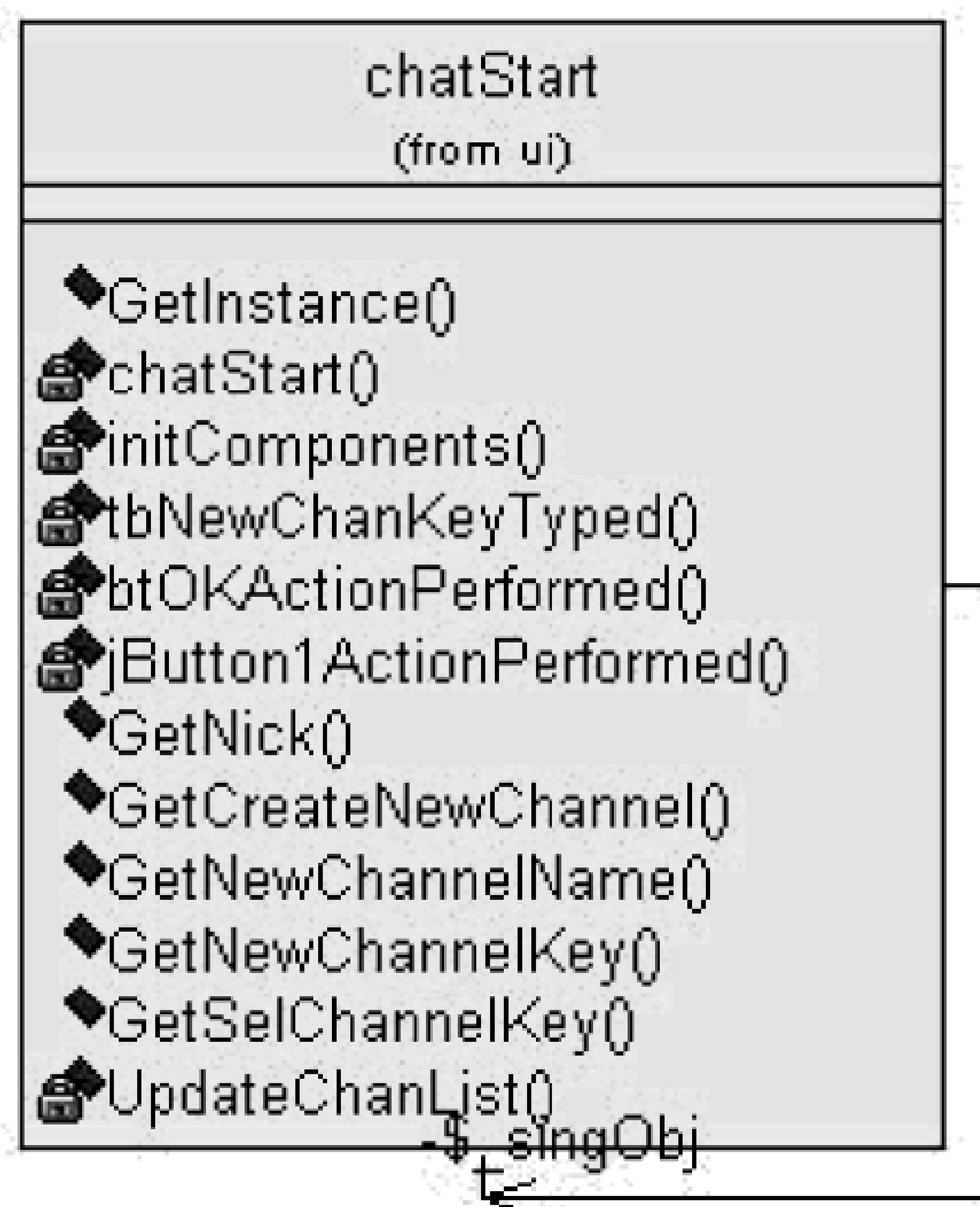


图 12.26　chatStart 类的类图

由图 12.26 所示的类图可以看出，它的所有方法都为实现以上的几个基本功能而设置的，如得到用户名字、创建一个新频道、得到新频道的名字、得到新频道的密码等，下面是它的实现源代码。

chatStart 类，用来实现用户登录或注册频道的功能，它是整个系统的启动界面。此类的源代码位于 p2pchat 工程的 p2p.chat.ui 包下。类的源代码请参考随书光盘，chatStart 类的位置如下：

【源代码：\ch12\ch12_code\p2pchat\src\p2p\chat\ui\chatStart.java】

在 chatStart 类中，主要完成以下几部分的工作：

- 类的初始化及声明。
- 启动界面的初始化。
- 处理界面中按钮的动作和功能。
- 更新当前频道列表。

那么围绕以上这几个工作，就需要相应的方法来实现。下面就分别对实现这几个功能的主要方法及实现步骤进行说明。

chatStart 类的初始化，主要完成相关包的引入、界面元素的定义、变量的声明等，实现方法如下：

```
/**
 * @author gl
 * chatStart 类，是系统界面模块的重要组成部分，主要用来实现用户登录或注册频道的界
 * 面，同时完成用户注册、登录或新建频道的功能，是整个系统启动的开始界面，继承自
 * JFrame 类。
*/
    //定义一个名为 chatStart 类，继承自 JFrame 类
public class chatStart extends JFrame {
    //以下是 chatStart 类作为启动界面，其界面控件的变量声明如下
    private JButton btOK;
    private JButton jButton1;
    private JLabel jLabel1;
    private JLabel jLabel2;
    private JLabel jLabel3;
    private JLabel jLabel4;
    private JLabel jLabel5;
    private JLabel jLabel7;
    private JLabel jLabelline;
    private JLabel jLabelline2;
    private JScrollPane jScrollPane1;
    private JList ltChans;
    private JTextField tbKey;
    private JTextField tbNewChan;
    private JTextField tbNickname;
    private JTextField tbSelKey;
    // 变量声明结束
    /*在本系统中，每个用户都可以单独地启动一个例程加入到网络中，因而设计成单例模式，当
    有用户需要登录里，只需启动一个 chatStart 类的实例即可，以下是单例模式的实现。*/
    private static chatStart singObj; //定义一个私有的、静态的 chatStart 类对象
    public static chatStart GetInstance(){
                                     //通过 GetInstance 返回一个 chatStart 对象
        if(singObj==null)            //判断当前的 chatStart 对象是否为空
            singObj=new chatStart(); //如果为空，则新建一个 chatStart 对象
```

```
        return singObj;              //如果不为空，将当前的 chatStart 对象返回
    }
    //私有的构造方法，保证不被其他类访问。此构造方法用来创建一个 chatStart 的界面实例
    private chatStart() {
     initComponents();               //通过 initComponents()方法初始化界面元素
     this.setBounds(400, 200, 280, 380);    //设置系统界面的大小
     this.update(this.getGraphics());       //调用 update 方法，更新当前界面
    }
```

以上的代码完成了 chatStart 类的声明、相关变量的定义，同时，也通过一个单例设计模式来获取 chatStart 类的实例对象。完成了以上的工作后，下面就需要对启动界面进行整体的设计。启动界面的初始化由一个名为 initComponents()的方法完成，主要用来实现界面元素的设计和布局，不涉及业务方法，具体实现如下：

```
/**
  * initComponents()方法，主要用于界面元素的设计和布局的，用到的都是 Java 图
  *形编程的知识，具体实现如下：
  */
 //定义一个 private 的 initComponents，只允许 chatStart 类访问
 private void initComponents() {
     //以下是对界面中出现的各个元素进行定义
     tbNickname = new JTextField();     //定义一个输入框，输入用户名称
     jLabel1 = new JLabel();            //定义文本标签 jLabel1
     jLabel2 = new JLabel();            //定义文本标签 jLabel2
     jScrollPane1 = new JScrollPane();  //定义滚动面板 jScrollPane1
     ltChans = new JList();             //定义列表，用来显示频道
     jLabel3 = new JLabel();            //定义文本标签 jLabel3
     jLabel4 = new JLabel();            //定义文本标签 jLabel4
     tbNewChan = new JTextField();      //定义一个输入框，输入频道名称
     jLabel5 = new JLabel();            //定义文本标签 jLabel5
     tbKey = new JTextField();          //定义一个输入框，输入登录密码
     btOK = new JButton();              //定义一个确定按钮 jLabel1
     jButton1 = new JButton();          //定义按钮 jButton1
     tbSelKey = new JTextField();       //定义一个输入框，用来设置密码
     jLabel7 = new JLabel();            //定义文本标签 jLabel7
     jLabelline = new JLabel();         //定义一条线，用于界面分隔
     jLabelline2 = new JLabel();        //定义一条线，用于界面分隔
     getContentPane().setLayout(null);  //设置界面的布局方式

     /*以下是对定义的各个组件进行初始化及相关值的设定*/
     //设置一个关闭按钮，此界面是可关闭的
     setDefaultCloseOperation (WindowConstants.EXIT_ON_CLOSE);
     //以下是对界面的标题、布局、背景色、大小等进行设定
     setTitle("P2P Chat 用户登录/注册");
     setAlwaysOnTop(true);
     setBackground(new java.awt.Color(255, 255, 255));
     setName("frmStart");
     setResizable(false);
     getContentPane().add(tbNickname);
     tbNickname.setBounds(80, 10, 180, 20);
     //对文字标签进行初始化
     jLabel1.setText("用户名称:");
     getContentPane().add(jLabel1);
     jLabel1.setBounds(10, 10, 70, 20);
```

```
//初始化一个分隔线，让整个界面看起来更有层次感
jLabelline2.setFont(new java.awt.Font("Tahoma", 1, 12));
jLabelline2.setText("***********************************");
getContentPane().add(jLabelline2);
jLabelline2.setBounds(10, 40, 260, 14);
//对频道操作区的相关初始化
jLabel2.setText("加入一个现有的频道:");
getContentPane().add(jLabel2);
jLabel2.setBounds(10, 55, 120, 14);
jLabel2.setForeground(new Color(255,0,0));
//用一个滚动面板显示当前已有的频道列表
jScrollPane1.setViewportView(ltChans);
getContentPane().add(jScrollPane1);
jScrollPane1.setBounds(10, 100, 250, 110);
jLabelline.setFont(new java.awt.Font("Tahoma", 1, 12));
jLabelline.setText("***********************************");
getContentPane().add(jLabelline);
jLabelline.setBounds(10, 245, 260, 14);
//用户创建一个新频道时的操作区
jLabel3.setText("创建一个新的频道:");
getContentPane().add(jLabel3);
jLabel3.setBounds(10, 260, 148, 14);
jLabel3.setForeground(new Color(255,0,0));
jLabel4.setText("频道名称:");
getContentPane().add(jLabel4);
jLabel4.setBounds(10, 280, 60, 20);
//以下是新建一个频道时的按钮动作
tbNewChan.addKeyListener(new KeyAdapter() {
    public void keyTyped(KeyEvent evt) {
        tbNewChanKeyTyped(evt);
    }
});
getContentPane().add(tbNewChan);
tbNewChan.setBounds(70, 280, 90, 20);
//用户设定密码
jLabel5.setText("密码:");
getContentPane().add(jLabel5);
jLabel5.setBounds(165, 280, 40, 20);
tbKey.setEnabled(false);
getContentPane().add(tbKey);
tbKey.setBounds(200, 280, 60, 20);
//“确定”按钮的初始化
btOK.setText("确定");
btOK.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent evt) {
        btOKActionPerformed(evt);
    }
});
getContentPane().add(btOK);
btOK.setBounds(90, 310, 80, 23);
//更新当前频道列表的操作
jButton1.setText("更新当前频道列表");
//用于更新频道列表的监听器
jButton1.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent evt) {
        jButton1ActionPerformed(evt);
    }
});
getContentPane().add(jButton1);
```

```
    jButton1.setBounds(10, 75, 150, 20);
    getContentPane().add(tbSelKey);
    tbSelKey.setBounds(160, 215, 100, 20);
    jLabel7.setText("加入此频道的认证密码:");
    getContentPane().add(jLabel7);
    jLabel7.setBounds(10, 215, 150, 20);
    //调用 pack()方法，将所有组件有序地结合在一起
    pack();
}//组件初始化结束
```

以上就是本系统中启动界面的实现方法，主要涉及的就是 Java 界面编程的相关知识，没有什么特别可讲解的地方，读者只需看懂源代码即可。

完成以上的工作以后，在 chatStart 类中，还需要处理界面中各种按钮的动作，也就是当单击“确定”按钮时，程序的逻辑是如何跳转的，需求所定义的功能又是如何实现的。下面就讲一下程序的实现步骤：

```
/**
 *当用户完成所有的输入，单击“确定”按钮时，用户就会完成登录进入到聊天网络，
 *以下的 btOKActionPerformed 方法就具体实现了这个按钮的动作。
 */
// btOKActionPerformed()方法，当用户单击“确定”按钮时将要执行的动作
private void btOKActionPerformed(ActionEvent evt) {
   //对用户的输入名称进行判断，不允许匿名登录
    if(GetNick().length()==0){
        //如果用户输入的名称为空，弹出提示信息，程序返回
        JOptionPane.showMessageDialog(this,"用户名称不能为空","错误",
        JOptionPane.ERROR_MESSAGE);
        return;
    }
    //对用户选择或创建频道进行判断，要么选择一个已有的频道，要么创建新的
    if(ltChans.getSelectedValue()==null && !GetCreateNewChannel()){
        //对不满足条件的输入行为，弹出提示信息
        JOptionPane.showMessageDialog(this,"你必须选择加入一个已有的频道或是
        创建一个新的频道","错误",JOptionPane.ERROR_MESSAGE);
        return;
    }
    //用户在加入频道的时候，需要检查用户名和频道列表，这个过程需一定的连接和缓冲时间，
      这时会弹出一个对话框，提示用户等待
    JDialog jWait=new JDialog(this,"请稍候... 检查你的用户名和当前可用的频道列
    表");
    jWait.setResizable(false);
    jWait.setAlwaysOnTop(true);
    jWait.setBounds(400, 200, 400, 20);
    jWait.setVisible(true);
    jWait.requestFocus();
    boolean NickAvail=true,ChanAvail=true;
    NickAvail=Manager.GetInstance().TrySetNick(GetNick());
    //如果此频道是用户新建的频道，则检查此频道是否空闲
    if(GetCreateNewChannel())
       ChanAvail=Manager.GetInstance().IsChannelFree(GetNewChannelName());
    jWait.dispose();
    //检查同一频道内的用户名称是否唯一
    if(!NickAvail){
        JOptionPane.showMessageDialog(this,"用户名称已经被使用了",
                                          "错误",JOptionPane.ERROR_MESSAGE);
        return;
```

```
        }
        //检查网络中的频道名称是否唯一
        if(GetCreateNewChannel()){
            if(ChanAvail){
                Channel.CreateNew(GetNewChannelName(),GetNewChannelKey());
                dispose();
            }
            else
                JOptionPane.showMessageDialog(this,"此频道已经存在了",
                                          "错误",JOptionPane.ERROR_MESSAGE);
        }
        else{
            String selChan=(String)ltChans.getSelectedValue();
            try {
                //当用户加入一个新频道时，需要对用户输入的频道密码进行认证
                Channel.JoinExisting(selChan,GetSelChannelKey());
                System.out.println("正在请求加入频道……");
                //由于系统是多线程实现的，所以需要 synchronized 关键字进行处理
                synchronized(Manager.GetInstance().WaitForJoinAck){
                Manager.GetInstance().WaitForJoinAck.wait(Manager.Default-
                OperTimeout);}
                if(Manager.GetInstance().GetCurrentChannel()==null){
                     //如果用户密码认证错误，弹出提示信息，并退出系统
                    JOptionPane.showMessageDialog(this,"连接超时(可能由于密码认证
                    错误)
                                             ","Error",JOptionPane.ERROR_MESSAGE);
                    System.exit(-1);
                }
                dispose();
            } catch (Exception ex) {
                ex.printStackTrace();
            }
        }
}
```

以上就是当用户在启动界面中单击确定按钮时所要执行的动作，在这些动作中，主要是完成对用户输入的合法性进行判断，然后根据输入的结果跳转到相应的逻辑操作。

chatStart 类中的主要方法，基本上就是以上所讲的几个，另外还需强调一点的就是，在启动界面中需要对当前的频道列表进行更新。因为用户登录的时候，可以选择当前已有的频道，那么更新频道列表，就是把当前存在于网络中的所有频道展示出来。以下就是更新频道列表方法的具体实现。

```
//更新频道列表的方法，定义为一个 private 的名为 UpdateChanList()的方法
    private void UpdateChanList(){
        ltChans.setModel(new AbstractListModel() {
            //通过 Manage 类实例的 GetAvailableChannels()方法，取得当前所有的频道
            String[] strings = Manager.GetInstance().GetAvailableChannels();
            //得到当前频道的个数
            public int getSize() { return strings.length; }
            //取出所有的频道，返回频道信息
            public Object getElementAt(int i) { return strings[i]; }
        });
    }
```

以上就是 chatStart 类的实现过程，此类主要用来对界面进行布局，对界面中的按钮、文本输入框、文本标签、列表框等进行初始化，同时对相关组件设置相应的触发事件的动

作，这些动作主要通过监听器来实现，重点是需要理解“确定”按钮触发时所要执行的动作。

chatStart 类中还涉及其他的一些辅助方法，读者要结合源代码来学习、理解。

2. 系统主界面

从登录/注册界面进入系统以后，就进入了 P2P 频道界面，这个频道界面类似于聊天室的界面，也是系统执行的主界面。因为 Peer 间的即时通信、文件共享、数据交互都是依赖于这个频道来实现的，这个界面的关闭也意味着 Peer 结点退出了当前频道。

频道界面也就是主界面是由 mainFrame.java 类来实现的，它是系统运行的主平台，即时通信的基本功能都依赖于这个界面来实现。从设计上讲，主界面应该有 3 个基本的区域。

（1）Peer 列表区域：用于展示 Peer 结点的信息，以列表的形式展示此频道中所有的 Peer 结点，同时还要提供对 Peer 结点的操作，如选择一个 Peer 进行会话、查看 Peer 的共享文件等。

（2）消息的发送和展示区域：本系统中的频道提供了类聊天室的会话环境，那么至少要有一个消息发送框和消息内容展示的区域。

（3）菜单列表区域：mainFrame 是系统的主界面，那么在这个主界面中就需要提供即时通信基本功能的入口，也就是需要一个区域来放置功能菜单，这些菜单的功能包括共享文件、发送文件、接收文件、清屏等。

根据以上要求，mainFrame 类的设计图如图 12.27 所示。

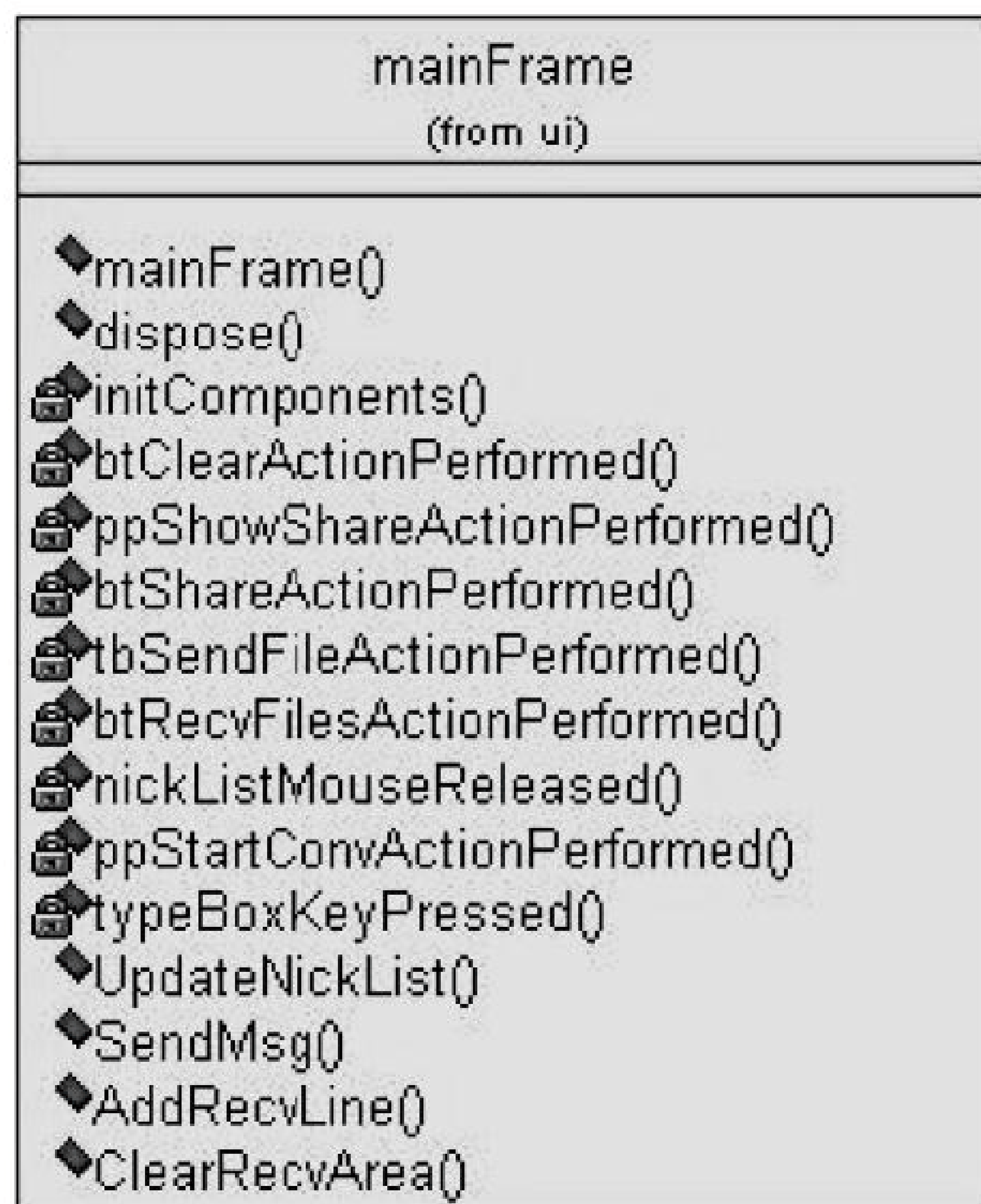


图 12.27　mainFrame 类的类图

由图 12.27 可以看出，mainFrame 类中大部分方法都是关于对界面控件进行监听操作的，这些方法可以让用户在单击相应的控件时执行相应的动作，以完成一系列的功能。图 12.27 的 ppShowSharActionPerformed()是显示共享列表的方法，tbSendFileActionPerformed()是发送文件方法，btRecvFilesActionPerformed()是接收文件的方法等，这些方法的具体实现请参阅此类的源代码。

mainFrame 类，是系统的主操作平台，它以频道操作界面为载体，完成用户的各种操作功能，也提供了执行其他功能模块的入口。此类的源代码位于 p2pchat 工程的 p2p.chat.ui 包下。请参考随书光盘，mainFrame 类的位置如下：

【源代码：\ch12\ch12_code\p2pchat\src\p2p\chat\ui\mainFrame.java】

在 mainFrame 类中，关于类的声明、变量的定义、界面初始化等方法，请读者自行参考源代码。下面主要对此类中所涉及的业务方法进行讲解，这些方法主要体现在对各个按钮的操作中，主要有：

- ❑ 显示当前频道中的共享文件。
- ❑ 将自己的文件共享出去。
- ❑ 发送文件。
- ❑ 接收文件。
- ❑ 向另一个 Peer 发起私人会话。
- ❑ 清屏操作。

下面就对实现以上几个功能的方法进行简要说明：

```
//以下几个方法是各个按钮动作的具体执行//
    //执行显示共享的动作，显示当前频道中，Peer 的共享文件
    private void ppShowShareActionPerformed(ActionEvent evt) {
     //确保选择了且选择的是一个 Peer 实例
     if(nickList.getSelectedValue()!=null && nickList.getSelectedValue()
     instanceof Peer)
      // 调用 peerShareFrame 的 ForUser()方法，将此 Peer 的共享文件展示出来
      peerShareFrame.ForUser((Peer)(nickList.getSelectedValue())).Show();
    }
    //执行主动共享的动作，当用户单击此按钮时，会将自己本地的目录共享出去
    private void btShareActionPerformed(ActionEvent evt) {
        //调用 Manager 类并取得一个实例，通过 SetMyShare()方法，以共享本地目录
        Manager.GetInstance().SetMyShare(btShare.isSelected());
    }
    //执行发送文件的动作，单击此按钮后，会让用户选择一个文件，然后发送出去
    private void tbSendFileActionPerformed(ActionEvent evt) {
        //调用 Manager 类并取得一个实例，通过 SendFileToAll()方法将文件发送出去
        Manager.GetInstance().SendFileToAll();
    }
     //执行接收文件的动作，单击此按钮后，会弹出接收文件的界面，显示 Peer 结点接收到的文
       件情况
    private void btRecvFilesActionPerformed(ActionEvent evt) {
       //调用 Manager 类并取得一个实例，通过 ShowRecvFiles()方法显示接收到的文件
       Manager.GetInstance().ShowRecvFiles();
    }
    //发起一个私人会话，右键选择 Peer，单击此按钮后，会弹出私人聊天的界面
    private void ppStartConvActionPerformed(ActionEvent evt) {
     //确保选择了且选择的是一个 Peer 实例
     if(nickList.getSelectedValue()!=null && nickList.getSelectedValue()
     instanceof Peer)
    //调用 Manager 类并取得一个实例，通过 StartPrivateConversation 发起一个点到点
      的私人会话
     Manager.GetInstance().StartPrivateConversation((Peer)(nickList.
     getSelectedValue()));
    }
    //执行清屏操作，当用户单击“清屏”按钮时，会将界面中消息框的内容置为空
```

```
    private void btClearActionPerformed(ActionEvent evt) {
        //通过 setText()方法，将消息框 msgArea 内的内容置为空
        msgArea.setText("");
    }
```

以上的这几个方法并不是独立存在的，它需要借助其他的类和方法来共同完成，尤其是 Manager 类（后文会讲到）。所以读者在学习这几个方法的时候，要能从全局的角度去理解、学习。

除了以上这几个方法外，mainFrame 类中还有几个比较重要的方法，对实现整个系统的功能也非常重要。

```
// nickListMouseReleased()方法，用户在用右键选择 Peer 时弹出菜单的操作
private void nickListMouseReleased(MouseEvent evt) {
    //进行判断，保证有 Peer 被选中
    if(evt.getButton()!=evt.BUTTON1)
        //以下的代码用来弹出右键单击后的操作选项
        nickPopup.show(nickList,evt.getX(),evt.getY());
}
//用来设定键盘，当在键盘上按下回车时，发送一条消息
private void typeBoxKeyPressed(KeyEvent evt) {
    //触发一个 KeyEvent 事件，并对键盘输入的字符进行判断
    if(evt.getKeyChar()=='\n')
        //如果是回车，则发送消息
        SendMsg();
}
// UpdateNickList()方法，用来更新 Peer 列表的操作
public void UpdateNickList(Peer[] iUsers){
    //对当前 Peer 列表的排列模式进行判断
    if(nickList.getModel()==null || !(nickList.getModel() instanceof
     NickListModel))
        //设定一个新的排列模式
        nickList.setModel(new NickListModel(iUsers));
        //在些排列模式的基础上更新 Peer 列表
        ((NickListModel)nickList.getModel()).Update(iUsers);

}
 //SendMsg()方法，用于发送一条消息的操作
 public void SendMsg(){
   //取得一个 Manager 类的实例，通过 SendChanMessage()方法，将输入框中的消息发送出去
   Manager.GetInstance().SendChanMessage(typeBox.getText());
   //消息发送完成后，将消息输入框的内容置为空
   typeBox.setText("");
 }
```

在 mainFrame 类中，还有一些其他方法，如用来显示主界面内容的 initComponents()方法，此方法完成界面组件的进行初始化功能，这些组件包括菜单项、文本框、列表框、弹出对话框等。其中，菜单项主要通过 JPopupMenu 类来实现，其中的每个菜单项则由 JMenuItem 对象来表示，文本框用来提供用户消息的输入，同时还用来显示系统消息和用户向整个频道发送的消息。列表框主要用来显示当前的 Peer 用户列表情况。弹出对话框是针对菜单项设置的，当用户单击菜单中相应的功能项时，就会弹出此功能的界面，这些功能界面由特定的类来实现。

关于 initComponents()方法的详细实现过程，所涉及的大部分都是 Java 界面编程的知识，请读者结合源代码自己学习，这里就不再讲解了。

3．文件共享界面

在主界面中，当用户选择一个 Peer 右建单击显示共享的时候，如果此 Peer 有共享文件，就会弹出一个显示文件共享的界面。当没有共享文件时，会先弹出提示信息，然后再弹出共享列表为空的文件显示界面，与文件共享有关的界面由 peerShareFrame 类来实现。

文件共享是本系统一个重要的功能，文件共享界面主要用来展示 Peer 结点共享的文件信息。这个共享信息包括两个方面，一个是显示共享的文件列表，以相对路径的形式列出所有的共享文件；另一个是显示文件的内容，当选中某一个共享文件的时候，会相应地显示出此文件的内容。当然，还可以在共享列表中选择一个或多个文件将其下载到本地。这是 peerShareFrame 类所要完成的基本功能，此类的设计结构图如图 12.28 所示。

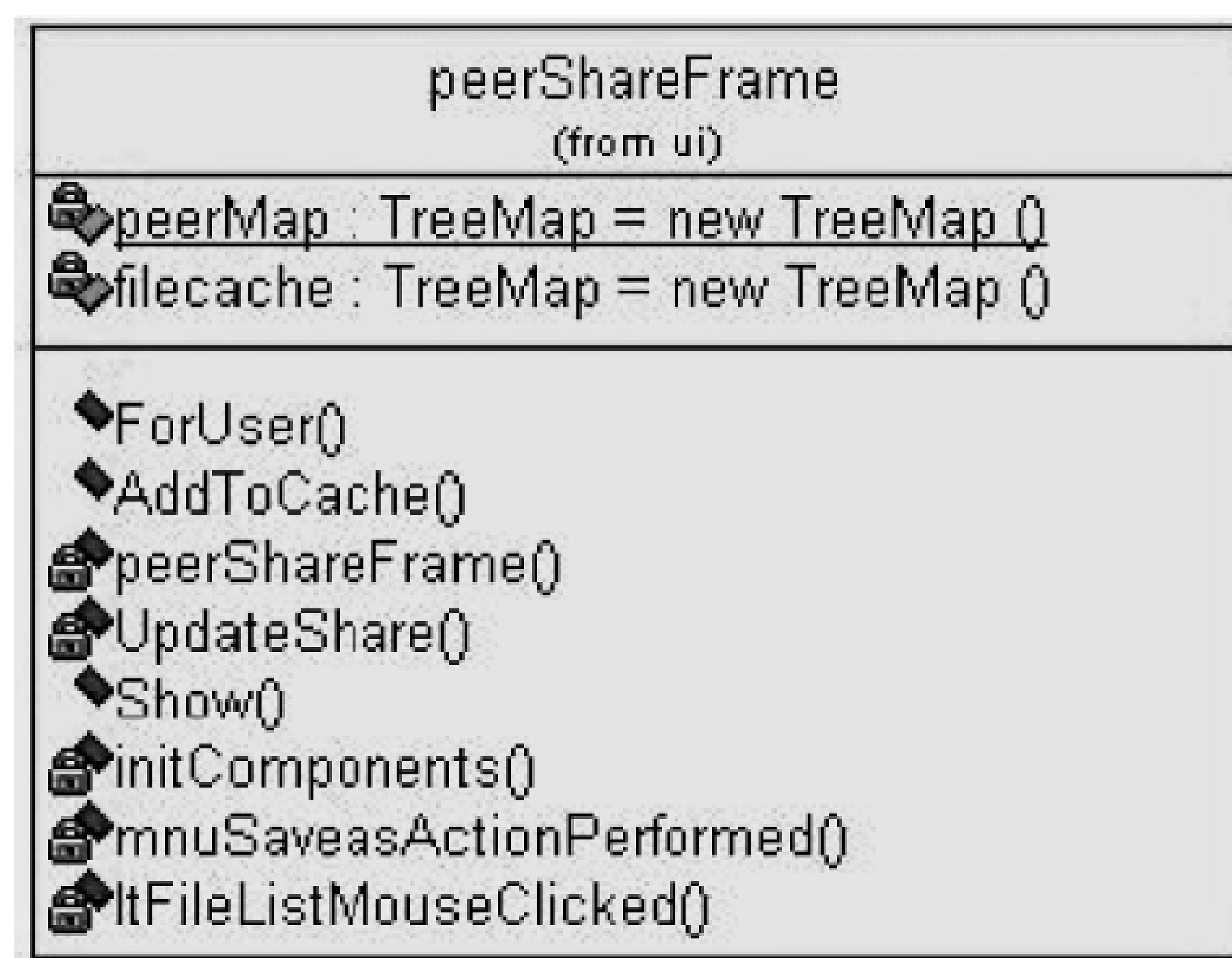


图 12.28 peerShareFrame 类的类图

在如图 12.28 所示的类结构中，有两个 Map 类型的数据结构，它们存储的都是<key，value>形式的键值对，其中一个是 peerMap，主要用来存储<peer，peerShareForm>对。这种存储信息的意思是，每一个 Peer 对应一个文件共享界面，针对多个不同的 Peer 可以打开多个文件共享界面，分别查看它们的共享信息面不会发生冲突，这些都由 peerMap 的键值对结构来控制。

另一个 Map 结构是 filecache，它所存储的是<String，AttachmentMessage>对，这种存储结构的意思是，一个文件名对应一个文件内容。也就是说，在 filecache 的结构中所存储的是文件的信息，通过文件名——文件内容的形式，得到了文件名（文件路径）也就能通过 Map 的索引得到相应的文件内容。这样，实现的功能就是：当用户单击共享列表中的某一文件名时，对应此文件名的文件内容就可显示出来。

注意：本系统中，通过文件名读取文件内容时，是以文本形式读取的，类似于记事本程序打开文件的效果，所以非文本格式的文件读出来后可能会是乱码。

在 peerShareFrame 类的设计中，主要完成以下几方面的功能：

- 是显示 Peer 共享文件信息的界面。
- 展示共享文件列表并显示文件内容。
- 将网络共享的文件存储到本地。

下面分别对以上这几个功能所涉及的方法进行讲解，peerShareFrame 类的源代码位于 p2pchat 工程的 p2p.chat.ui 包下，参考随书光盘，类文件的位置如下：

【源代码：\ch12\ch12_code\p2pchat\src\p2p\chat\ui\peerShareFrame.java】

首先讲解一下，上文提到的两个 Map 结构的定义及具体的使用方法。

```
/**
 *以下是存储<Peer,peerShareFrame>值对的 Map 结构的定义及声明，使用 TreeMap
 *的结构表示，Key 值存储的是 Peer 结点的信息，Value 值存储的是文件共享信息
 *通过<Key, Value>对的形式，保证每一个 Peer 对应一个文件共享的界面。
 */
private static TreeMap peerMap=new TreeMap();
/**
 *以下是存储<String,AttachmentMessage>值对的 Map 结构的定义及声明，同样也
 *使用 TreeMap 的结构表示，Key 值存储的是 String 类型的文件名信息，Value 值
 *存储的是 AttachmentMessage 类型的文件内容信息，保证一个文件名对应着自身
 *的文件内容。
 */
private TreeMap filecache=new TreeMap();
// ForUser()方法，传入一个 Peer 对象，返回一个 peerShareFrame 对象
public static peerShareFrame ForUser(Peer iPeer){
    //对当前的 peerMap 进行判断，是否已有此 Peer 对应的共享信息
    if(!peerMap.containsKey(iPeer))
        //如果 Key 值中没有此 Peer，则在 peerMap 新加入此 Peer 的共享信息
        peerMap.put(iPeer,new peerShareFrame(iPeer));
    //否则，根据 Key 值，取出用 Value 表示的此 Peer 的共享信息
    peerShareFrame retFrm=(peerShareFrame)peerMap.get(iPeer);
    //将得到结果的共享信息更新
    retFrm.UpdateShare();
    //返回一个表征共享信息的 peerShareFrame 对象
    return retFrm;
}
//AddToCache()方法，用来表示共享的文件内容信息
public void AddToCache(AttachmentMessage iMsg){
    //将要共享的文件名、内容存入到 filecache 中
    filecache.put(iMsg.GetFileName(),iMsg);
    //选择了其中某一个文件，并查看此文件的内容
    if(ltFileList.getSelectedValue()!=null  && ltFileList.getSelected-
    Value().equals(iMsg.GetFileName())){
        tbContent.setText(iMsg.GetContent());
    }
}
```

以上是对 peerShareFrame 类中两个重要数据结构的讲解，接着再讲一下，如何在文件共享界面中显示并更新当前的文件共享列表。

在界面中显示共享文件列表，就是将用户的共享文件名列出来的过程，在本类中用名为 UpdateShare()的方法表示更新并显示用户共享文件列表的过程。

```
//UpdateShare 方法，用于查看并更新用户共享文件信息，返回一个 Boolean 类型的值
    private boolean UpdateShare(){
        //对用户当前的共享情况进行判断，调用 Manager 类的 GetUserShare()方法
        if(Manager.GetInstance().GetUserShare(peer)==null){
            //当用户没有共享时，弹出相应的提示信息
            JOptionPane.showMessageDialog(this,"当前没有共享文件","错误",
            JOptionPane.ERROR_MESSAGE);
            //因为没有共享信息，共享界面设置不可见
            setVisible(false);
            //返回 false，说明当前用户没有共享文件
```

```
            return false;
        }
    //如果用户共享文件不为空，调用 Manager 类的 GetUserShare()方法取出共享信息
    ltFileList.setModel(new StringListModel(Manager.GetInstance().
    GetUserShare(peer)));
    //返回 true，说明当前有共享文件
    return true;
    }
```

在 UpdateShare()方法中，核心是调用了 Manager 类 GetUserShare 的实现，整个逻辑比较简单易懂，这里就不再赘述。

注意：在 UpdateShare()方法中，有一个 setModel()方法，此方法表示的是对当前文件列表的文件进行排序模式，比如按文件名还是文件大小等，读者可查看源代码，可重写这个方法，自己设定排序模式。

在用户共享的界面中，还有一个操作功能是，当用户选择某一文件后，如何查看此文件名对应的文件内容，在本系统的实现中，用户选择某一文件名后，只需用鼠标双击此文件名，就可查看文件内容了，具体由 ltFileListMouseClicked()方法来实现，代码如下：

```
/**
 * ltFileListMouseClicked()方法，是一个针对鼠标双击的动作而执行方法，主要用来显示
 *文件的内容，当鼠标双击时，触发动作，然后程序读取文件显示给用户
 */
private void ltFileListMouseClicked(MouseEvent evt) {
        //判断鼠标的动作
        if(evt.getButton()!=evt.BUTTON1)
            //显示选中的文件名
            mnuPopup.show(ltFileList,evt.getX(),evt.getY());
        //对选中的共享文件是否存在进行判断
        if(ltFileList.getSelectedValue()==null)
                return;
        //如果当前的 filecache 结构中，不包含当前选中的文件名则进行如下操作
        if(!filecache.containsKey(ltFileList.getSelectedValue())){
           //系统产生服务信息，向所有用户广播，提示此文件不存在
            ServiceMessage newMsg=new ServiceMessage(Manager.GetInstance().
            GetMe(),peer,ServiceMessage.CODE_ASK_FILE,(String)ltFileList.
            getSelectedValue());
            // 通过网络模块的 DispatchToAll()方法，将消息广播出去
            Manager.GetInstance().GetDispatcher().DispatchToAll(newMsg);
        }else{
//如果此文件存在，则在 tbContent 框中，将文件内容显示给用户
tbContent.setText(((AttachmentMessage)filecache.get(ltFileList.getSelec
tedValue())).GetContent());
        }
    }
```

在 peerShareFrame 类中还有一个重要的方法，就是将用户的共享文件下载到本地，这个功能在执行的过程中，用户选择某一文件后右击，会弹出“另存为”按钮，再单击此按钮就会执行下载及保存过程，此方法的实现如下。

```
/**
 * mnuSaveasActionPerformed()方法，也是一个针对鼠标动作的执行方法，主要用来实
 * 现将用户选择的共享文件存储到本地的功能
```

```
*/
    //定义一个 private 类型的 mnuSaveasActionPerformed()方法，传入 ActionEvent 对象
    private void mnuSaveasActionPerformed(ActionEvent evt) {
        //调用 JList 对象的 getSelectedIndex 获取共享文件列表中选中项的索引
        if(ltFileList.getSelectedIndex()>=0){
            //判断 filecache 结构中是否存储有选中的文件
            if(!filecache.containsKey(ltFileList.getSelectedValue()))
                //如果没有，程序返回
                return;
            //如果存在，则根据 FileDialog 取得一个 File 对象
            java.io.File oFile=FileDialog.SaveFileDialog();
            //判断得到的 File 对象是否为空
            if(oFile==null)
                return;
          try {
                //通过 Map 结构的 get 方法，从 filecache 中取得文件内容
                AttachmentMessage curSel=
                    (AttachmentMessage)filecache.get(ltFileList.get-
                    SelectedValue());
                //直接调用 AttachmentMessage 对象的 SaveToFile()方法，将文件保存
                curSel.SaveToFile(oFile.getPath());
            //捕获并处理异常
            } catch (Exception ex) {
                //如以上过程发生错误，则弹出对话框，提示错误信息
                JOptionPane.showMessageDialog(this,ex.getMessage(),"
                                                错误",JOptionPane.ERROR MESSAGE);
                ex.printStackTrace();
            }
        }
```

以上就是 peerShareFrame 类中几个重要方法的说明，还有一个界面组件初始化的方法，请读者自行参考源代码学习。

在即时通信系统中，文件共享是一个重要的功能，读者要着重理解文件共享的过程和实现逻辑。

4．接收文件界面

在主界面中，当用户单击“接收文件”按钮时，就会弹出一个与接收文件操作有关的界面，此界面就是接收文件的界面，主要用来展示并处理来自其他 Peer 发送过来的文件。

接收文件界面与文件共享界面类似，也有 3 个基本功能，即显示接收的文件列表、展示文件的内容和将接收到的文件存储到本地。如果某一个 Peer 要接收的文件列表里显示为空，说明当前没有需要接收的文件。接收文件的界面由 recvFilesFrame 类来完成，在设计上也很简单，如图 12.29 所示的就是 recvFilesFrame 类的设计图。

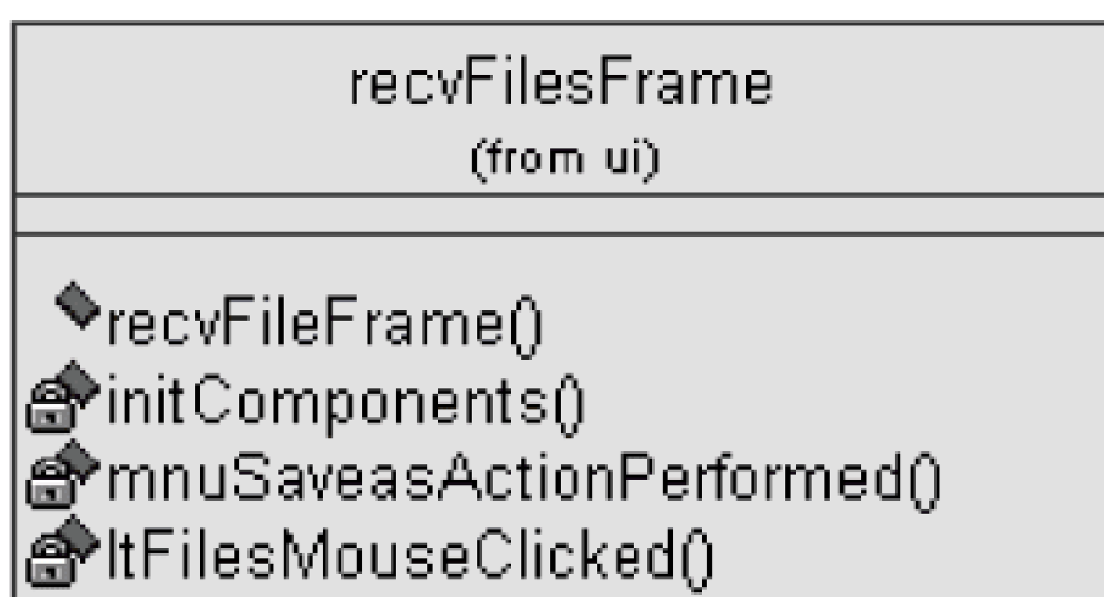


图 12.29 recvFilesFrame 类的类图

从图 12.29 中就可以看出，recvFilesFrame 类的结构和方法都很简单，一个构造方法和一个初始化方法，mnuSaveActionPerformed()方法是用来完成通过右键将接收到的文件存储到本地的操作。而 ltFilesMouseClicked()方法，则是鼠标在单击一个文件的文件名时，在另一区域对应显示此文件的内容。

recvFilesFrame 类的源代码位于 p2pchat 工程的 p2p.chat.ui 包下，读者可参考随书光盘，类文件的位置如下：

【源代码：\ch12\ch12_code\p2pchat\src\p2p\chat\ui\recvFilesFrame.java】

由于 recvFilesFrame 类与 peerShareFrame 类的主要方法及功能类似，主要就是完成查看文件内容和将接收到的文件存储到本地的操作。这些操作方法及实现过程在上文的 peerShareFrame 类中已经讲过，所以关于 recvFilesFrame 类，请读者参考源代码学习，这里不再单独讲解。

5．私人会话的聊天界面

Peer 结点间的点对点会话，是 P2P 即时通信系统的核心内容。在本系统中，用户可以在系统主界面窗口中进行操作，以发起一个私人会话。当用户选择 Peer 列表中的某一个 Peer 右击，在弹出的快捷菜单中选择“私人聊天”的时候，就会弹出私人聊天界面，此界面用来完成两个 Peer 之间的私人会话功能。

私人聊天界面主要用来完成两个 Peer 结点之间的消息交互，整个界面的布局很简单，只需一个消息输入框和会话内容的展示框。用户从输入框里输入要会话的消息，按下回车后消息发送出去，Peer 间的聊天内容会同步的显示在内容展示框里。此外，还有一个小功能是，当用户在输入框里输入“/cls”或“/clean”时，会执行清屏操作，自动清空消息展示框里的聊天内容。私人聊天的界面，由 privateConvFrame 类来实现，此类的类图如图 12.30 所示。

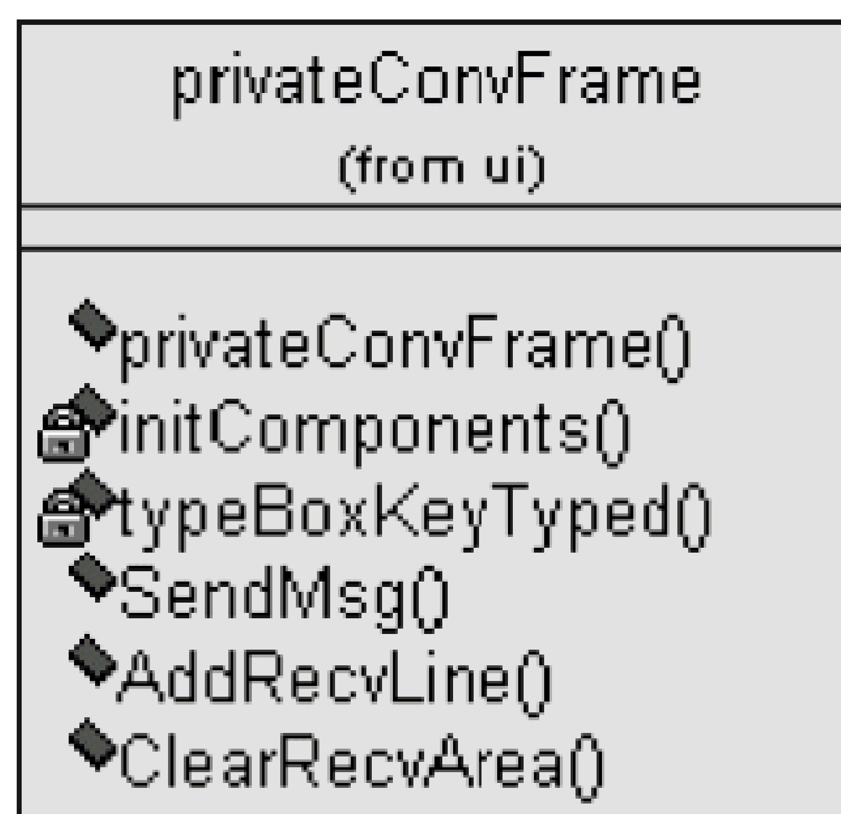


图 12.30　privateConvFrame 类的类图

由图 12.30 所示，privateConvFrame 只有几个方法，这些方法主要就是实现发送消息、接收消息、清屏等操作。其中 AddRecvLine()方法，就是在每条接收到的消息后面加上“/n”，用来实现各条消息之间的换行操作。

privateConvFrame 类的源代码位于 p2pchat 工程的 p2p.chat.ui 包下。读者可参考随书光盘，类文件的位置如下：

【源代码：\ch12\ch12_code\p2pchat\src\p2p\chat\ui\privateConvFrame.java】

下面就对 privateConvFrame 类的具体实现进行说明。

```
/**
 * privateConvFrame 类，用来完成两个 Peer 间私人会话界面的初始化、点到点之间消
 *息的交互功能，通过此界面，可以发送一条消息到一个指定的 Peer 结点，也可以接
 *收并展示来自此结点的消息。
 */
package p2p.chat.ui;                    //程序所在包路径的声明
import java.awt.*;                      //引入 Java 的 awt 包，与执行按钮动作有关
import javax.swing.*;                   //引入 Java 的 swing 包，与界面初始化有关
import p2p.chat.fun.PrivateConversation;   //引入系统功能模块的私人会话功能
//声明一个 privateConvFrame 类，继承自 JFrame 类
public class privateConvFrame extends JFrame {
    //定义一个私有的 PrivateConversation 变量，私人会话的核心都由此类实现
    private PrivateConversation refConv;
    //私人会话界面中用到的组件元素声明
    private JScrollPane jScroller;          //提供可滚动显示的窗口
    private JTextArea msgArea;              //显示消息内容的消息域
    private JTextField typeBox;             //输入会话消息的文本框
    //变量声明结束

    /**构造方法 privateConvFrame()，传入 PrivateConversation 对象作为参数，用来
    创建一个新的私人会话界面，*/
    public privateConvFrame(PrivateConversation iRef) {
        refConv=iRef;                           //将传入的参数赋值给 refConv 变量
        initComponents();                       //初始化界面组件
        this.setBounds(300, 300, 300, 300); //初始界面大小
        setVisible(true);                       //设置可见
        requestFocus();                         //使此界面获得焦点
        setTitle(refConv.GetTo().GetName());//设置界面标题
        typeBox.requestFocus();                 //消息输入框自动获得焦点

    }
   //私人会话的聊天界面组件初始化
    private void initComponents() {
        typeBox = new javax.swing.JTextField();      //初始化一个消息输入框
        jScroller = new javax.swing.JScrollPane();  //初始化一个滚动视图
        msgArea = new javax.swing.JTextArea();       //初始化消息展示框
        //设置标题
        setTitle("私人聊天");
        //设置名称
        setName("frmPrivateConv");
        //监听键盘按键动作
        typeBox.addKeyListener(new KeyAdapter() {
            public void keyTyped(KeyEvent evt) {
                typeBoxKeyTyped(evt);
            }
        });
        //设置整个界面的布局，在 ContentPane 添加并布局消息输入框
        getContentPane().add(typeBox, BorderLayout.SOUTH);
        //设置消息展示框的颜色
        msgArea.setBackground(new java.awt.Color(244, 241, 241));
        //设置消息展示框可显示消息的列、每行显示的字数
        msgArea.setColumns(20);
        //设置消息展示框的内容是不可编辑状态
```

```
        msgArea.setEditable(false);
        //设置消息展示框可显示行数
        msgArea.setRows(5);
        //通过 jScroller，设置消息展示框是可滚动的，可通过滚动条显示
        jScroller.setViewportView(msgArea);
        //将整个消息展示框添加到 ContentPane 上方的部分
        getContentPane().add(jScroller, java.awt.BorderLayout.CENTER);
        //pack()方法，将各个组件打包，合理紧促地显示出来
        pack();
    }
    //typeBoxKeyTyped()方法，处理键盘按键的动作，当按下回车后，发送消息
    private void typeBoxKeyTyped(KeyEvent evt) {
        //对按键字符进行判断，是否是回车
        if(evt.getKeyChar()=='\n')
            //如果按下的是回车，通过 SendMsg()方法，将消息发送出去
            SendMsg();
    }
    //SendMsg()方法，是发送消息的具体实现方法，在执行过程中，还需要对输入的消息进行判
      断，当消息内容是/clear 或者/cls 时，则是执行清屏操作的命令，而不是发送消息
    public void SendMsg(){
        //对消息输入框的内容时行判断，是否是/clear 或者/cls
        if(typeBox.getText().equals("/clear") || typeBox.getText().equals
        ("/cls") ){
            //如果是，则执行 ClearRecvArea()方法，执行清屏操作
            ClearRecvArea();
        }else
            //否则执行 PrivateConversation 类的 SendMessage()方法，将消息发送出去
            refConv.SendMessage(typeBox.getText());
        //一条消息发送完成后，将当前的消息输入置为空
        typeBox.setText("");
    }
    //AddRecvLine()方法，对每条接收到的消息后加上\n，用于自动换行
    public void AddRecvLine(String iMsg){
        //在 msgArea 的消息展示框中加入接收到的消息内容，并加上\n 符换行
        msgArea.setText(msgArea.getText()+iMsg+"\n");
        //通过 scrollRectToVisible()方法，超过 20 字长时将字段向右滚动
        msgArea.scrollRectToVisible(new Rectangle(0,msgArea.getHeight()
       -20,1,1));
    }
    //ClearRecvArea()方法，执行清屏操作的功能
    public void ClearRecvArea(){
        //将消息展示框内的内容置空
        msgArea.setText("");
    }
}
```

以上就是私人会话界面的初始化、聊天消息的发送、接收等方法的具体实现，在学习的过程中，最好能结合源代码实际上机操作学习。

6．文件选择对话框

文件选择对话框，并不是一个真正意义上的界面，只是在系统应用中方便用户进行目录和文件选择的一个显示对话框。

当用户单击共享目录、发送文件或存储文件时都会弹出这个对话框，用户可以在这个

对话框中定位目录和文件，也可以选择存储的路径，此功能由 FileDialog 类来实现。

借助 javax.swing 包下的 JDialog 类和 JFIleChooser 类，可以轻松地实现一个弹出文件选择对话框的操作，这是一个封装的操作，与本系统中其他的类无关。

FileDialog 类，主要用来弹出一个文件选择的对话框，方便用户操作。此类的源代码位于 p2pchat 工程的 p2p.chat.ui 包下。读者可参考随书光盘，类文件的位置如下：

【源代码：\ch12\ch12_code\p2pchat\src\p2p\chat\ui\FileDialog.java】

以下就是 FileDialog 类的主要实现过程及方法。

```
 /**
  * FileDialog 类，在本系统中，当用户进行与文件选择或存储有关的操作时，会弹出
  *此对话，在此对话框中用户可以浏览本地文件目录或路径。
  */
package p2p.chat.ui;                             //声明类文件所在的包路径
import java.awt.event.*;                         //引入 awt 有关的类，与动作执行有关
import java.io.File;                             //引入 File 类，与文件操作有关
import javax.swing.*;                            //引入 swing 包，与界面布局操作有关

//定义一个 FileDialog 类，主要用于弹出一个文件对话框，提供用户流浏览文件、选择目录、
选择路径等操作
public class FileDialog {
    //定义一个 private 静态的 File 类型变量
    private static File selfile;
    //GenericFileDialog()方法，方法体中调用与之同名的另一个重写的方法
    private static File GenericFileDialog(int iType){
        return GenericFileDialog(iType,JFileChooser.FILES ONLY);
    }
    /**
     * 另一重写的 GenericFileDialog()方法，通过调用 javax.swing 包下的 JDialog 类和
     * JFileChooser 类来实现相关操作，返回一个 File 类的对象，表示虚拟的文件路径
     *或文件目录，以下代码的实现有相对固定的模式，读者自行学习，不再细讲。
     */
    private static File GenericFileDialog(int iType,int iSelMode){
        selfile=null;
        final JDialog fDiag=new JDialog();
        fDiag.setModal(true);
        final JFileChooser fc=new JFileChooser();
        fc.setDialogType(iType);
        fc.setFileSelectionMode(iSelMode);
        fc.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent evt){
                if(evt.getActionCommand().equals("ApproveSelection"))
                    FileDialog.selfile=fc.getSelectedFile();
                    fDiag.setVisible(false);
                }
        });
        fDiag.getContentPane().add(fc);
        fDiag.setDefaultCloseOperation(JDialog.HIDE ON CLOSE);
        fDiag.setBounds(300, 300, 400, 300);
        fDiag.setVisible(true);
        return fc.getSelectedFile();
    }
 //public static 类型的 DirFileDialog ()方法，实现目录文件对话框的功能，根据
```

```
FileChooser.OPEN_DIALOG 参数和 JFileChooser.DIRECTORIES_ONLY 参数，返回
GenericFileDialog()方法得到的 File 对象
    public static File DirFileDialog(){
          return GenericFileDialog(JFileChooser.OPEN_DIALOG ,
                                  JFileChooser.DIRECTORIES_ONLY);
    }
    // public static 类型的 OpenFileDialog()方法，实现打开文件对话框的操作，调用
GenericFileDialog()方法，根据 JFileChooser.OPEN_DIALOG 参数返回一个 File 对象
    public static File OpenFileDialog(){
        //调用 GenericFileDialog()方法，传入 JFileChooser.OPEN_DIALOG 参数
        return GenericFileDialog(JFileChooser.OPEN_DIALOG);
    }
//public static 类型的 SaveFileDialog()方法，用于保存文件时打开文件对话框的操作，
调用 GenericFileDialog()方法，根据 JFileChooser. SAVE_DIALOG 参数返回一个 File
对象
    public static File SaveFileDialog(){
        return GenericFileDialog(JFileChooser.SAVE_DIALOG);
    }
}
```

以上就是 FileDialog 类的全部实现过程，此类是一个工具类，与系统的业务功能没有什么联系，是系统可用性、完善性的必要辅助。由于相对独立，FileDialog 类可以完整地移植到任何其他的与文件选择操作有关的系统中。

12.5.7　全局管理类及 main()方法的实现

以上讲解了系统的主要类的设计和源代码，根据对系统架构的分析，还需一个全局的管理类来实现以上所有类的调度和管理，这个类就是 Manager.java。要使系统运行起来还需要一个 Main.java 的主类，它是 main()方法所在的类，系统的执行就从运行 main 类开始，本节就讲一下如何开发 Manager 和 main 类。

1. Manager全局管理类的实现

上文在讲系统设计中其他类的时候，很多地方都提到 Manager 类，此类简单地说就是整个系统的管理中枢，很多其他的类都要与它发生联系。Manager 在英文单词中是“管理者”的意思，而此处的 Manager 类就是一个针对系统的全管理类，以单例模式实现，主要用于其他各类之间的衔接和调用，Manager 类的类图如图 12.31 所示。

Manager 类在系统中起到了全局的管理作用，所以这个类的设计比较复杂，方法也很多，涉及很多核心方法的实现，此类的源代码位于 p2pchat 工程的 p2p.chat 包下。读者可参考随书光盘，类文件的位置如下：

【源代码：\ch12\ch12_code\p2pchat\src\p2p\chat\Manager.java】

Manager 类作为一个全局的管理类，要实现的功能很多，所有的方法不可能一一讲解，本文中只对 Manager 类所完成的一些核心方法进行说明。Manager 类中所涉及的主要方法及操作大致可以分为以下几类。

（1）功能操作方法：功能操作，是指 Manager 类所实现的几个基本功能，如显示接收的文件表、发送文件、开起私人会话、取得用户共享信息等，这些功能也是由 Manager 类来完成。

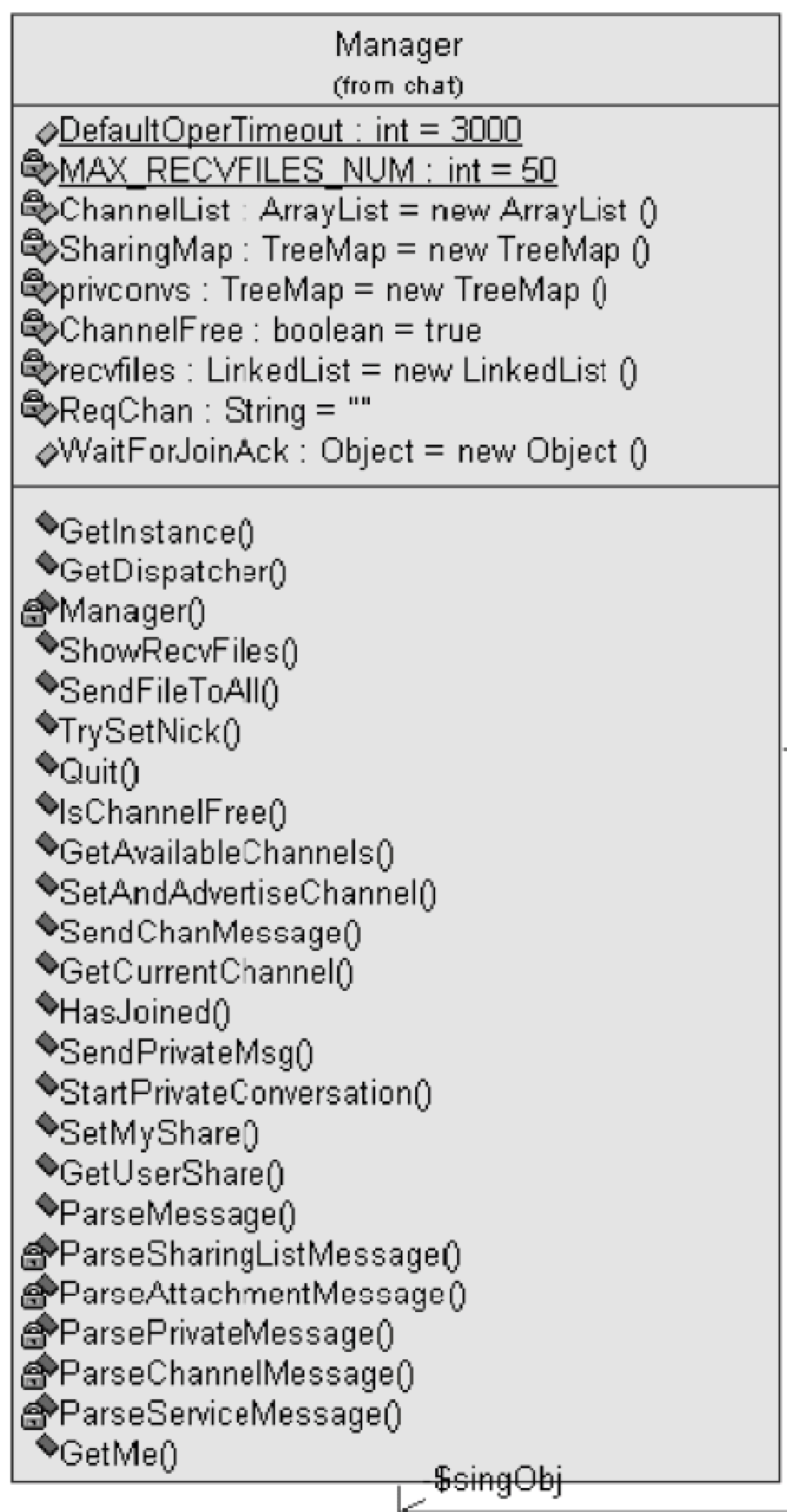


图 12.31　Manager 类的类图

（2）系统设置和结果判定方法：系统设置和结果判定这一类方法，主要是 Manager 用来完成与系统设定和判断有关的操作，如设置一个 Peer 的名字、判断当前频道是否空闲、设置消息广播的形式等，也都由 Manager 类完成。

（3）消息处理方法：消息处理是 Manage 类的重要职责，每一个用户进入 P2P 频道以后，都会有一个 Manager 进行监视，用户的所有动作，都会通过消息的形式反馈出来。比如用户加入了、用户退出了、用户共享文件了、用户接收文件了、用户发消息了、系统出错了等，这些消息都由 Manager 类会根据用户的操作进行反应，并及时将不同类型的消息发布出去。

下面分别对 Manager 类中的这 3 类方法进行说明。

在讲解这 3 类方法之前，首先从设计模式的角度来说一下 Manager 类的设计。上文已经说过，Manager 类是一个单例模式的类，这种模式的好处就是可以保证每一个 p2pChat 的系统进程中只有一个 Manager 类实例。当用户进入 P2P 聊天系统后，就会得到这个 Manager 实例，然后 Manager 就会发挥其总调度师的功能，协调其他的类共同完成即时通信的功能。下面首先说一下 Manager 类中单例模式的实现。

```
/**
 * Manager 类中单例模式的实现过程
 */
    //定义一个私有的、静态的 Manager 变量 singObj
    private static Manager singObj;
```

```
    //定义一个公有的、静态的 GetInstance()方法，返回一个 Manager 实例
    public static Manager GetInstance() {
      try {
          //判断当前的 Manager 对象 singObj 是否为 null
          if(singObj==null)
              //如果为 null，通过 new 操作符产生一个 Manager 对象
              singObj=new Manager();
      //捕获并处理异常
      } catch (Exception ex) {
          ex.printStackTrace();
      }
      //将 Manager 对象 singObj 返回
      return singObj;
  }
  //定义一个私有的 Manager 构造方法，保证其他的类无法产生 Manager 对象
  private Manager() throws Exception{
      //在构造方法中对 MulticastDispatcher 对象和 Peer 对象赋值
      Dispatcher=new MulticastDispatcher();
      me=Peer.Anonymous;
  }//构造单例模式结束
```

以上就是 Manager 类单例模式的实现过程，单例模式是 Java 设计模式中最简单也最常用的模式之一，这里就不展开讲解了，有兴趣的读者可自己学习相关资料。

下面讲一下 Manager 类中与功能操作有关的方法，代码如下：

```
/**
 * StartPrivateConversation()方法：用于开启一个私人会话，传入 Peer 对象为参数
 */
//定义一个 StartPrivateConversation，参数指定了要给哪个 Peer 发送消息
public void StartPrivateConversation(Peer iTo){
    //对要发送消息的 Peer 进行判断，保证存在
    if(iTo==null)
        return;
    //判断 TreeMap 结构中的 Key 值中是否包含当前的 Peer
    if(!privconvs.containsKey(iTo)){
        //如果不存在此对象，则新添加到 privconvs 中
        privconvs.put(iTo,new PrivateConversation(iTo));
    }else
        //否则将此 Peer 值对应的私人会话界面打开，开起一个私人会话进程
     ((PrivateConversation)privconvs.get(iTo)).Show();
}

/**
 * GetUserShare()方法，用来取得用户的共享信息，传入 Peer 对象作来参数
 */
// GetUserShare()方法，参数指定了要取出哪个 Peer 的共享信息，返回 String 数组
public String[] GetUserShare(Peer iUser){
    //判断 SharingMap 结构的 Key 中，是否包含当前 Peer 的值
    if(!SharingMap.containsKey(iUser))
        //如果不存在，说明当前用户没有共享文件
        return null;
   //如果存在，则从 SharingMap 中取出此用户的共享文件，以 String 数组形式返回
   return (String[])SharingMap.get(iUser);
}
/**
 * ShowRecvFiles()方法，显示用户接收的文件列表，在某一个 P2P 频道内，当有 Peer
```

```
 * 将自己的文件共享出去以后，同一个频道内其他所有 Peer 都可以接收到此共享文
 * 件的信息，本方法就是用来显示接收到的文件列表信息的。
 */
public void ShowRecvFiles(){
  //定义一个 AttachmentMessage 数组，每个 AttachmentMessage 对象表示一个文件
  AttachmentMessage []recvArr=new AttachmentMessage[recvfiles.size()];
    try {
        //对用户接收信息 recvFilesFrame 对象进行判断，是否为 null
        if(recvui!=null)
            //调用 dispose()方法进行处理
            recvui.dispose();
        //将接收的到文件信息转换成数组
        recvfiles.toArray(recvArr);
        //通过一个窗口来展示接收到的文件信息
        recvui=new recvFilesFrame(recvArr);
    //捕获并处理异常
    } catch(Exception ex) {
        ex.printStackTrace();
}
}

/**
 * SendFileToAll()方法，用来在同一个 P2P 频道内将文件分发给所有的 Peer，具体实
 * 现过程如下：
 */
public void SendFileToAll(){
    try {
     //先弹出文件选择对话框，用户选择一个用于发送的文件
        File chFile=FileDialog.OpenFileDialog();
        //对选择的 File 对象进行判断
        if(chFile==null)
            return;
        //判断所选文件是否存在
        if(!chFile.exists())
            return;

      //判断所选的文件是否符合规定的大小
      if(chFile.length()>GetDispatcher().GetMaxFileSize()){
      //本系统中，发送文件的最大容量 65000 字节，如果超出这个容量则弹出错误的提示信息
      JOptionPane.showMessageDialog(null,"文件太大
                        (最大容量为: "+ GetDispatcher().GetMaxFileSize()
                                +" bytes)","错误",JOptionPane.ERROR_MESSAGE);
            //程序返回
            return;
        }
      //调用 AttachmentMessage 构造方法，产生一个 AttachmentMessage 对象
      AttachmentMessage newMsg=new AttachmentMessage(GetMe(),chFile,false);
      //调用 DispatchToAll()方法，将文件对象分发出去
      GetDispatcher().DispatchToAll(newMsg);
    //捕获并处理异常信息
    } catch (Exception ex) {
        JOptionPane.showMessageDialog(null,ex.getMessage(),"
                                        错误",JOptionPane.ERROR_MESSAGE);
        ex.printStackTrace();
    }
}
/**
```

```
     * Quit()方法，用于用户退出 P2P 即时通信系统的操作，当执行此方法时，系统正
     * 常退出用具体实现过程如下：
     */
    public void Quit(){
        //判断当前频道是否为空，如果不为空，说明还有其他 Peer 在线
        if(curChan!=null){
          //通过 ServiceMessage 对象产生一条通告消息
          ServiceMessage newMsg=newServiceMessage(
                      GetMe(),ServiceMessage.CODE PART,curChan.GetName());
        //通过 DispatchToAll 将此消息分发出去，告诉其他所有 Peer：“我退出了”
        GetDispatcher().DispatchToAll(newMsg);
        }
        //系统正常结束
        System.exit(0);
    }
```

以上的几个方法，都是与系统的功能操作有关的方法，实现了 P2P 即时通信系统中诸如共享、文件分发、会话、系统退出等基本功能。以下要讲的是，与系统的设置及判定有关的方法，对整体系统功能的实现也非常重要。

```
/**
 * TrySetNick()方法，传入一个 String 类型的参数，返回一个 Boolean 型的值，用
 * 于设定 Peer 的名称，如果能成功设定则返回 true，否则返回 false
 */
public boolean TrySetNick(String iNick){
    //根据构造方法，取得一个自身信息的 Peer 对象
    me=new Peer(iNick);
    // 通过 SetStatus()方法，设定状态，传入的参数表示状态信息
    me.SetStatus(Peer.STATUS_ASKINGNICK);
    //通过 ServiceMessage 对象封装一条消息
    ServiceMessage newMsg=new ServiceMessage(
           Peer.Anonymous,ServiceMessage.CODE_QUERY_NICK_FREE,iNick);
    //将此消息通过 DispatchToAll()方法分发出去
    GetDispatcher().DispatchToAll(newMsg);
    try {
        //让线程暂停执行
        Thread.sleep(DefaultOperTimeout);
    //捕获并处理异常
    } catch (InterruptedException ex) {}
    //如果 Peer 设定名称的请求失败
    if(me.GetStatus()==Peer.STATUS_NICKFAILED)
        return false;
    //设定 Peer 状态为 Peer.STATUS AUTH
    me.SetStatus(Peer.STATUS_AUTH);
    return true;
}
/**
 * IsChannelFree()方法，传入一个 String 类型的频道名称作为参数，返回一个 Boolean
 * 型的值，用判定当名为输入参数的频道是否空闲，如果空闲返回则返回 true，
 * 否则返回 false
 */
public boolean IsChannelFree(String iChanName){
    //设置 Boolean 型的 ChannelFree 默认值为 true
    ChannelFree=true;
    //通过 ServiceMessage 对象，封装一条查询当前频道是否空闲的系统消息
    ServiceMessage newMsg=new ServiceMessage(
```

```
            GetMe(),ServiceMessage.CODE_QUERY_CHAN_FREE,iChanName);
    //通过 DispatchToAll()方法，将此消息发布出去
    GetDispatcher().DispatchToAll(newMsg);
    try {
        //线程暂停执行
        Thread.sleep(DefaultOperTimeout);
    //捕获并处理异常
    } catch (InterruptedException ex) {}
    return ChannelFree;
}
/**
 * GetAvailableChannels()方法，用于获取当前已存在的频道列表，返回一个 String
 * 类型的数组表示已有的频道，具体实现如下：
 */
public String[] GetAvailableChannels(){
    //初始化一个 outArr 字符串数组
    String []outArr=new String[ChannelList.size()];
    //将 ChannelList 对象存储到此数组中，表示当前的频道列表
    ChannelList.toArray(outArr);
    //返回包含结果信息的 outArr 数组
    return outArr;
}
/**
 * SetMyShare 方法，用来设置自身的共享信息，传入一个 Boolean 型的参数，用
 * 来表示是否设定共享，具体实现如下：
 */
public void SetMyShare(boolean iSet){
    //对传参进行判断，是否需要设定共享
    if(!iSet){
        //调用 Unshare()方法，取消共享
        myshare.Unshare();
    }else{
        //否则，打开一个文件选择对话框，选择需要共享的文件
        File shareFld=FileDialog.DirFileDialog();
        //对共享的目录进行判断，需要满足 3 个条件，即目录不为 Null，是存在的，且是目录结构
        if(shareFld==null || !shareFld.exists() || !shareFld.isDirectory())
            return;
        //调用 ShareDir()方法，将选择的目录共享出去
        myshare.ShareDir(shareFld.getAbsolutePath());
       //系统打印一条消息，说明一下有多少个文件被共享出来了
       System.out.println(myshare.GetSharedFiles().length + " 个文件已被共享");
       //将当前 Peer 的共享情况信息封装一条 SharingListMessage 消息
       SharingListMessage newMsg=new SharingListMessage(
                                  GetMe(),myshare.GetSharedFiles());
       //DispatchToAll()方法将当前 Peer 的共享情况广播给频道内的其他 Peer
       GetDispatcher().DispatchToAll(newMsg);
    }
}
/**
 * SetAndAdvertiseChannel()方法，用于设定关广播频道信息的方法，传入一个
 * Channel 类型的对象作为参数具体实现如下：
 */
public void SetAndAdvertiseChannel(Channel iChan){
    //将传入的参数赋值给变量 curChan
    curChan=iChan;
    //在 curChan 中加入自身的 Peer 信息
    curChan.AddPeer(GetMe());
```

```
    ReqChan="";
    //对当前的广告线程进行判断
    if(advThread!=null)
        advThread.stop();
    //将 curChan 作为参数，新建一个 ChanAdvertiserThread 对象
    advThread=new ChanAdvertiserThread(curChan);
    //启动多线程，开始进行广播
    advThread.start();
}
```

以上的几个方法，与系统的一些参数设定、状态判断有关，主要用来辅助整个 P2P 即时通信系统的运行。

在 Manager 类中，还有一类非常重要的方法是进行消息处理的。在本文所开发的这个即时通信系统中，消息处理贯穿整个系统的各个环节，比如用户登录的时候需要知道当前网络中的频道信息、加入频道以后系统会向网络发出通知消息、运行过程中要共享文件、发起会话、设置共享等，都需要进行消息的处理，包括退出系统的时候，系统也会向其他还在线的 Peer 发出一句提示消息。所有这些消息都由 Manager 类进行统一封装，然后在各个不同的类、方法之间进行调用。

下面就讲一下 Manager 类中对消息的处理方法。

```
/**
 * SendChanMessage()方法，用于发送频道消息，如果涉及与频道有关的操作，
 * 就会发送此消息，此消息包含频道自身的一些信息，具体实现如下：
 */
public void SendChanMessage(String iMsg){
    //判断当前频道是否为空
    if(curChan==null)
        return;
    //通过 ChannelMessage 对象封装一条频道消息
    ChannelMessage newMsg=new ChannelMessage (iMsg,GetMe(),curChan);
    //调用 DispatchToAll()方法，将此消息发布出去
    GetDispatcher().DispatchToAll(newMsg);
}
/**
 * SendPrivateMsg()方法，用于发送私人会话消息，需要传入要发送的 Peer 对象，
 * 和消息内容，主要用到两个 Peer 之间的私人会话操作上，具体实现如下：
 */
public void SendPrivateMsg(Peer iTo,String iMsg){
    //通过 PrivateMessage 对象封装一条私人会话的消息
    PrivateMessage newMsg=new PrivateMessage(iMsg,GetMe(),iTo);
    //调用 DispatchToAll()方法，将此消息发布出去
    GetDispatcher().DispatchToAll(newMsg);
}
```

以下是处理各种消息的方法。在下面将要讲述的方法中，所有的方法都有一个明显的特征，方法名都是以 parse 单词开头，表示这个方法是“处理”的意思，它们是 P2P 即时通信系统中对各种消息进行处理的核心实现过程。

```
    /**
     * ParseSharingListMessage()方法，传入一个 SharingListMessage 对象作为参数，用
     * 于处理共享列表消息的方法，共享列表的有无、共享列表中文件的变化，都由
     * 此方法负责处理，具体实现如下：
     */
```

```
private void ParseSharingListMessage(SharingListMessage iMsg){
    //将传入的共享列表信息存储到 SharingMap 对象的 Map 结构中
    SharingMap.put(iMsg.GetSender(),iMsg.GetShareList());
    //判断当前的 P2P 频道是否为 null
    if(curChan!=null)
        //调用当前频道的 Notice()方法，通知当前频道中正在共享的文件信息
        curChan.Notice(iMsg.GetSender() + " 正在共享 "+ iMsg.GetShare-
        List().length +" 个文件… 在 Peer 列中右击可查看其共享信息");
}
 /**
 * ParseAttachmentMessage()方法，传入一个 AttachmentMessage 对象作为参数，用
 * 于处理文件信息的方法，共享文件、接收文件等，都由此方法负责处理
 */
private void ParseAttachmentMessage(AttachmentMessage iMsg){
    //判断是否有关于文件信息的请求
    if(iMsg.IsRequested()){
        peerShareFrame.ForUser(iMsg.GetSender()).AddToCache(iMsg);
    }else{
        //在接收到的文件中添加此消息
        recvfiles.add(iMsg);
        //判断接收到的文件大小是否条例规范
        if(recvfiles.size()>MAX RECVFILES NUM)
            //如果文件大小超界，则移除此文件
            recvfiles.removeFirst();
        //如果当前频道不为空
        if(curChan!=null)
            //频道发出通知消息，显示接收到文件的信息
            curChan.Notice("接收到新的文件 " + iMsg);
    }
}
/**
 * ParsePrivateMessage()方法，传入一个 PrivateMessage 对象作为参数，用来处理
 * Peer 之间的私人会话消息
 */
private void ParsePrivateMessage(PrivateMessage iMsg){
    //判断消息发送的对象，如果消息发到自身，则程序返回
    if(!iMsg.GetTo().equals(GetMe()))
        return;
    //通过 StartPrivateConversation，启动一个私人会话进程
    StartPrivateConversation(iMsg.GetSender());
    //处理 Peer 间的私人会话过程
    ((PrivateConversation)privconvs.get( Msg.GetSender())).Message-
    Arrival(iMsg.GetText());
}
/**
 * ParseChannelMessage()方法，传入一个 ChannelMessage 对象作为参数，用来处理
 * 与频道操作有关的各类消息
 */
private void ParseChannelMessage(ChannelMessage iMsg){
    //判断传入的 ChannelMessage 对象是否为空
    if(iMsg.GetChannel()==null)
        return;
    //判断 ChannelMessage 对象中的频道是否当前频道
    if(!iMsg.GetChannel().equals(curChan))
        return;
    iMsg.GetChannel().MessageReceived(iMsg);
```

```
    }

    /**
     * ParseServiceMessage()方法，传入一个 ServiceMessage 对象作为参数，用来处理
     * 与整个系统操作有关的各类服务性的消息
     */
    private void ParseServiceMessage(ServiceMessage iMsg){
        //判断此 ServiceMessage 消息是否广播、是否发向自身
        if(!iMsg.IsBroadcast() && !iMsg.GetToUser().equals(GetMe()))
            return;
        //通过 switch 操作对 ServiceMessage 的消息类型进行判断
        switch(iMsg.GetCode()){
            //如果是频道广播消息
            case ServiceMessage.CODE CHAN ADV:{
                //判断频道列表中是否包含当前的频道
                if(!ChannelList.contains(iMsg.GetArg())){
                    //在频道列表中加入频道消息
                    ChannelList.add(iMsg.GetArg());
                    //打印系统提示信息，输出当前网络中的频道名称
                    System.out.println("发现网络中有新的频道，名称为" +iMsg.
                    GetArg());
                }
            }
                break;
                //当有 Peer 加入频道的时候，处理 Peer 加入频道的消息
            case ServiceMessage.CODE JOIN:{
                //判断当前频道是否为 Null
                if(curChan==null){
                    ReqChan=iMsg.GetArg();
                }else{
                    //判断 ServiceMessage 中的频道与当前频道名是否相同
                    if(!iMsg.GetArg().equals(curChan.GetName()))
                        return;
                    //将 ServiceMessage 对象中的 GetSender()方法加入到当前频道
                    curChan.Join(iMsg.GetSender());
                     //将当前加入频道的信息，封装一个 ServiceMessage 对象向每一个发
                       送一个 Hello 消息，来通知：“我加入这个频道了”
                    ServiceMessage newMsg=new ServiceMessage(GetMe(),
                           ServiceMessage.CODE HELOJOIN,curChan.GetName());
                     //DispatchToAll()方法，向所有 Peer 客户端广播加入频道的消息
                     GetDispatcher().DispatchToAll(newMsg);
                    //判断当前 Peer 是否此频道的创造者
                    if(GetMe().equals(curChan.GetOwner())){
                      //将频道创建者的消息封装一个 ServiceMessage 对象
                      newMsg=new ServiceMessage(GetMe(),
                      ServiceMessage.CODE CHAN OWNER,curChan.GetName());
                        //DispatchToAll()方法，将此消息广播出去
                        GetDispatcher().DispatchToAll(newMsg);
                    }
                }
            } break;
            //当 Peer 加入频道后，处理系统的通知消息
            case ServiceMessage.CODE_HELOJOIN:{
                //对当前频道进行判断
                if(curChan==null && ReqChan.length()>0 &&
                                        ReqChan.equals(iMsg.GetArg())){
                    //系统打印消息，提示此 Peer 成功加入到频道
```

```
                System.out.println("已经成功加入到频道");
                //设置一个广播消息
                SetAndAdvertiseChannel(new Channel(iMsg.GetArg()));
                //在当前频道频道添加这个新加入的 Peer
                curChan.AddPeer(GetMe());
                synchronized(WaitForJoinAck){ WaitForJoinAck.notify();}
            }else if(curChan==null && ReqChan.length()<=0)
                return;
            if(! iMsg.GetArg().equals(curChan.GetName()))
                return;
            //将 Peer 加入此频道的消息发布出去
            curChan.AddPeer(iMsg.GetSender());
        }break;
         //处理当 Peer 离开频道时的消息
        case ServiceMessage.CODE PART:{
            //对当前频道进行判断
            if(curChan==null) return;
            if(!iMsg.GetArg().equals(curChan.GetName()))
                return;
            //将 Peer 离开频道的消息发布出去
            curChan.Part(iMsg.GetSender());
        }break;
        //处理频道创建者的消息
        case ServiceMessage.CODE_CHAN_OWNER:{
            //判断当前频道是否为 Null
            if(curChan==null) return;
            //判断 ServiceMessage 中的频道信息是否当前频道
            if(!iMsg.GetArg().equals(curChan.GetName()))
                return;
            //将此频道创建者的消息发布出去
            curChan.SetOwner(iMsg.GetSender());
        }break;
        //查看 Peer 共享文件时的系统消息
        case ServiceMessage.CODE ASK FILE:{
            //取得共享文件列表信息
            String []MyShareList=myshare.GetSharedFiles();
            //系统打印当前共享的文件信息
            System.out.println("共享的文件 : " +iMsg.GetArg());
            //如果共享文件列表中不存在文件，则程序返回
            if(MyShareList.length<=0)
                return;
            if(Arrays.binarySearch(MyShareList,iMsg.GetArg())<0)
                return;

            //得到共享的文件对象
            File sFile=new File(myshare.GetFullFilePath(iMsg.GetArg()));
            //判断文件是否存在
            if(!sFile.exists()){
                //如果文件不存在，系统打印提示信息
                System.out.println("找不到文件 " + sFile.getPath());
                return;
            }
            AttachmentMessage newMsg;
            try {
                //得到表示共享文件的 AttachmentMessage 对象消息
                newMsg = new AttachmentMessage(GetMe(), sFile, true);
                //在此消息中设置文件的名字
                newMsg.SetFileName(iMsg.GetArg());
```

```
                    //将关于文件共享的信息发布到频道中
                    GetDispatcher().DispatchToAll(newMsg);
                //捕获并处理异常信息
                } catch (Exception ex) {
                    ex.printStackTrace();
                }
            }break;
        …
}
```

以上就是处理各种消息的具体实现方法，还有其他一些方法，由于其实现过程比较简单，就直接省略了说明。读者需要结合源代码来理解这些方法的实现过程，这对理解整个系统的实现原理非常有帮助。

以下这么多处理方法，在实际的使用中，还需要一个总的调度方法，根据不同的消息类型执行不同的操作，这一功能由 ParseMessage()方法实现，以下就是实现过程。

```
/**
 * ParseMessage()方法，传入一个总的父类 Message 对象作为参数，对消息的不同
 * 类型进行判断，然后调用相应的业务逻辑去具体实现
 */
public void ParseMessage(Message iMsg){
    //如果消息对象是 ServiceMessage 实例
    if(iMsg instanceof ServiceMessage)
        //调用 ParseServiceMessage()方法，对 iMsg 进行处理
        ParseServiceMessage((ServiceMessage)iMsg);
    //如果消息对象是 ChannelMessage 实例
    else if(iMsg instanceof ChannelMessage)
        //调用 ParseChannelMessage()方法，对 iMsg 进行处理
        ParseChannelMessage((ChannelMessage)iMsg);
    //如果消息对象是 PrivateMessage 实例
    else if(iMsg instanceof PrivateMessage)
        //调用 ParsePrivateMessage()方法，对 iMsg 进行处理
        ParsePrivateMessage((PrivateMessage)iMsg);
    //如果消息对象是 AttachmentMessage 实例
        else if(iMsg instanceof AttachmentMessage)
            //调用 ParseAttachmentMessage()方法，对 iMsg 进行处理
            ParseAttachmentMessage((AttachmentMessage)iMsg);
        //如果消息对象是 SharingListMessage 实例
    else if(iMsg instanceof SharingListMessage)
       //调用 ParseSharingListMessage()方法，对 iMsg 进行处理
        ParseSharingListMessage((SharingListMessage)iMsg);
}
```

通过 ParseMessage()方法，有效地管理了各个方法之间的调用，针对系统中的每一个消息都可以做出合理的判定和恰当的处理，这样，整个程序的逻辑也就很清楚了。

Manager 类，是整个系统的核心类，集中协调和管理了整个系统中其他各类之间的关系。理解这个类，对理解整个系统的实现流程至关重要，所以读者需要结合源代码认真学习并领会。

2. main类的实现

main 类是系统的执行入口，只有一个 main()方法，方法体是一个多线程的代码段。执行过程也很简单，取一个 Manager 实例，然后通过 Manager 调用界面模块里的 chatStart 类，这样用户启动程序后就进入了登录/注册的界面。

在 main 类中用到了 ava.awt 包下的 EventQueue 类，EventQueue 是一个与平台无关的类，它将来自于底层同位体类和受信任的应用程序类的事件列入队列。

它封装了异步事件指派机制，该机制从队列中提取事件，然后通过对此 EventQueue 调用 dispatchEvent(AWTEvent)方法来指派这些事件（事件作为参数被指派）。该机制的特殊行为是与实现有关的。

main 类，是系统的主类，所有程序的执行入口。此类的源代码位于 p2pchat 工程的 p2p.chat 包下。读者可参考随书光盘，类文件的位置如下：

【源代码：\ch12\ch12_code\p2pchat\src\p2p\chat\Main.java】

main 类的实现过程及主要代码说明如下：

```
package p2p.chat;
 /*
 *Main 类，系统 main()方法所在的类，是整个系统的执行入口，只包括一个 main()方法
 */
import java.awt.EventQueue;
import p2p.chat.ui.chatStart;
public class Main {
      //主类，系统的入口函数
    public static void main(String[] args) {
       //调用 EventQueue 对象的 invokeLater()方法，传入多线程对象，使 runnable 的
         run()方法在 EventQueue 的指派线程中被调用
       EventQueue.invokeLater(new Runnable() {
           public void run() {
             //取得一个 Manager 的实例
               Manager.GetInstance();
               //启动用户登录注册界面，进入 P2P 即时通信系统
               chatStart.GetInstance().setVisible(true);
           }
       });
    }
}
```

注意：在以上的 main 类中，用到了 EventQueue 对象，此对象是一个与平台无关的类，它将来自于底层同位体类和受信任的应用程序类的事件列入队列。它封装了异步事件指派机制，该机制从队列中提取事件，提取的要求有两点，一是按顺序指派，也就是说，不允许同时从该队列中指派多个事件。二是指派顺序与它们排队的顺序相同，也就是说，如果 AWTEvent A 比 AWTEvent B 先排入到 EventQueue 中，那么事件 B 不能在事件 A 之前被指派。这是 Java 编程的一个知识点，读者了解一下即可。

main 类开发完成以后，运行 main 类的 main()方法，就可以启动整个系统的运行了。至此，整个系统所有类的设计、分析、源代码都全部讲解完毕了。

以上对整个系统的开发、系统中各个类的设计及源代码都进行了详细的说明，需要注意的是，系统的设计和开发并不是一步到位的，而是一个波浪式的上升过程。在实际开发过程中类的设计也并不是严格按照以上的顺序进行的，开发后面的功能有可能需要补充前面类的代码，而且整个开发是围绕着开发→测试→开发→再测试这样的过程不停反复进行的，开发完成后还要进行界面上的修饰、功能上的完善、Bug 的处理等，这样，才能最终

形成一个完整的系统。

12.6 系统测试与功能验证

系统开发完毕后，就是进行相关的测试工作，本系统是一个基于 P2P 的即时通信系统，要测试其运行情况和 Peer 间的通信，最好在两台以上的多台主机上进行。本文为简单地说明问题，就在两台主机上进行测试，有兴趣的读者可在局域网环境下的多台主机上进行测试。在测试的时候要注意和上文对类的设计与分析联系起来，要理解哪个类是实现哪些功能的。

注意： 本系统在单机上可以同时运行多个实例，这样在单机上也可以测试，但是单机上运行多个实例时，某些功能无法体现，建议读者最好不要在单机上测试。

12.6.1 系统的启动与简单消息交互

在 Eclipse 中运行 Main.java 类，就会启动系统的运行，系统运行后会弹出一个用户登录/注册界面，如图 12.32 所示。

（1）用户首先得输入自己的名称，然后可以有两个选择，一个是加入一个现有的频道，另一个是新建一个频道。因为这是第一个 Peer，所以当单击“更新当前频道列表”时，没有显示任何频道信息，因此，这里选择创建一个新的频道。如图 12.33 所示，新建一个频道，名称为 ChanneA，密码为 passwrod，用户名为 PeerA。

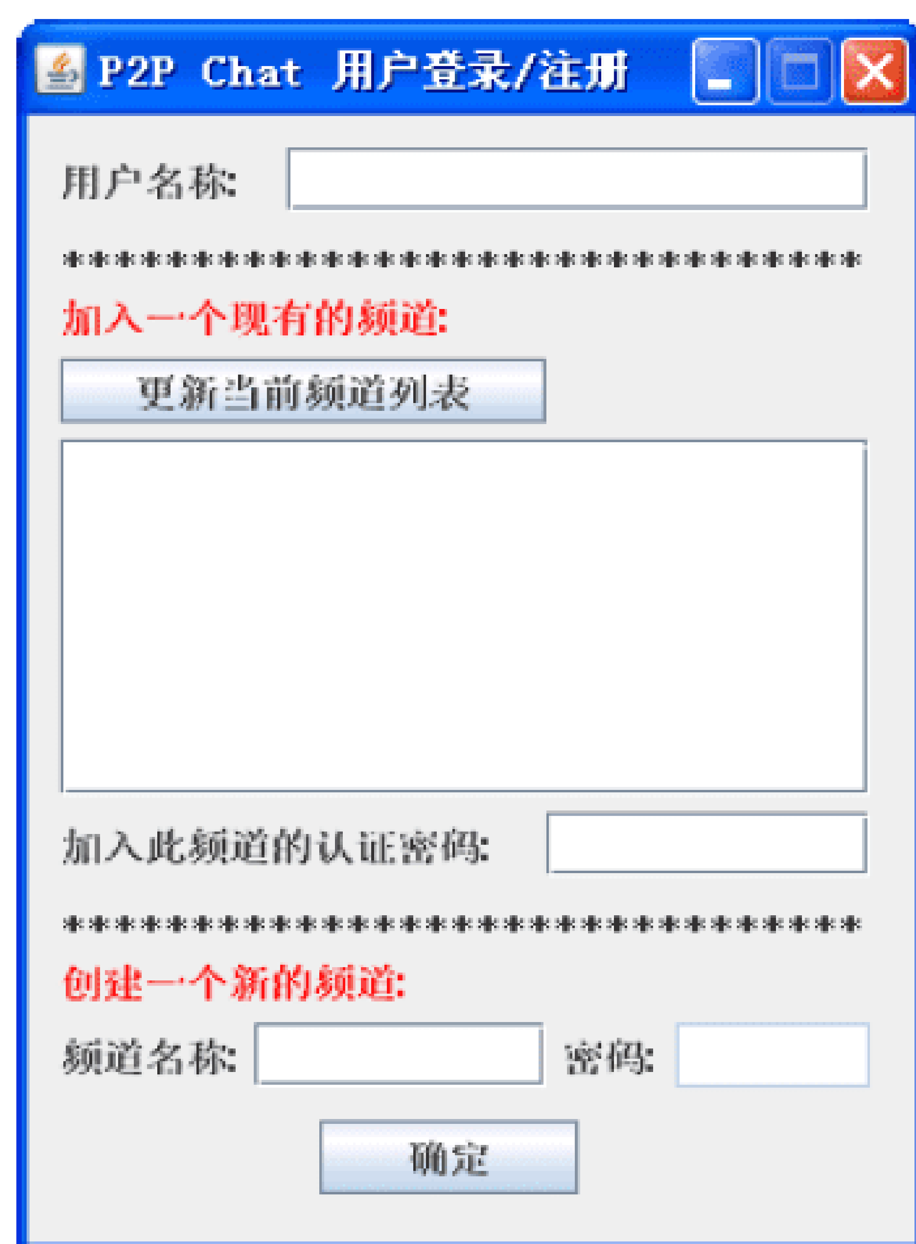


图 12.32 系统的登录注册界面



图 12.33 PeerA 创建一个新频道登录示意图

（2）在图 12.33 所示的界面中单击“确定”按钮，中间会有一个简短的连接、缓冲时间，然后一个名为 ChannelA 的 P2P 频道就创建好了，如图 12.34 所示。

注意： 如有一个频道创建好了之后，在 Eclipse 的控制也会打印出相应的通知消息，这种系统的通知消息就由上文所讲的消息类中 ServiceMessage 功能来实现。

（3）图 12.34 所示的就是一个新创建的 P2P 频道，这个频道中只包含一个名为 PeerA（自身）的结点信息，在这个频道界面中，可以发送消息并显示消息内容，如图 12.35 所示。



图 12.34　系统的主频道界面



图 12.35　在主频道界面中发送消息

名为 ChannelA 的频道创建以后就可以让其他的 Peer 来加入了，为了测试另一个 Peer 的加入，需要在另一台主机上运行本系统，为了与 PeerA 区分，命名此主机为 PeerB，在 PeerB 中运行系统后，也会弹出“登录/注册”界面，但在频道列表中就可以发现网络中已经创建的 ChannelA 频道了，如图 12.36 所示。输入用户名称 PeerB，然后选择已有的 ChannelA 频道，再输入 ChannelA 频道的认证密码 password，那么 PeerB 就可以加入 ChannelA 了，加入后在 PeerB 主机上显示的界面如图 12.37 所示。



图 12.36　另一个 Peer 登录已有频道时的界面



图 12.37　PeerB 登录主频道后的显示界面

（4）此时，ChannelA 频道的 Peer 列表中就同时出现了 PeerB 和 PeerA，证明 ChannelA 频道下目前有 PeerA 和 PeerB 两个结点。与此同时，在 PeerA 机器上，也会自动地更新 Peer 列表，并在聊天窗口中显示出通知消息。如图 12.38 所示，是 PeerA 机器上 ChannelA 频道的截图。

这样，PeerA 和 PeerB 两个结点，在没有服务器的情况下，借助 P2P 频道建立了连接，

它们之间就可以相互发送公共的会话消息。如图 12.39 所示，是 PeeA 与 PeerB 通过 ChannelA 频道交互消息的过程。



图 12.38　PeerA 主机中的主频道界面显示情况



图 12.39　在同一频道内两个 Peer 的消息交互

图 12.39 是在 PeerA 主机上的截图，在聊天窗口中，既显示结点间会话的消息，也显示系统的通知消息。单击上方菜单中的“清屏”按钮，就可以清空聊天窗口中的内容。

以上测试的就是系统的启动运行和 Peer 结点间简单通信。在这个过程中需要注意一点，在演示过程中，ChannelA 频道是由 Peer 创建的，但它并不属于 PeerA，当 PeerA 退出后，ChannelA 频道依然存在，只有当 ChannelA 频道中所有的结点都退出时，ChannelA 频道才会退出。如图 12.40 所示为 PeerA 退出后的情况 ChannelA 频道情况。



图 12.40　PeerB 主机上在 PeerA 退出频道后的界面显示

图 12.40 中，系统消息会显示 PeerA 离开了 ChannelA 频道，此时 ChannelA 频道依然存在，但只剩下一个 PeerB 结点，这也从另一个侧面说明了本系统中 P2P 的特性。

12.6.2　文件共享功能的测试

文件共享指的是，当一个 Peer 结点共享了某一目录后，其他所有结点都可以访问此

Peer 的共享目录，并可以将此共享目录的文件下载下来。

为测试需要，再加入一个名为 PeerC 的结点，同时共享 PeerB 主机上的一个目录，单击频道界面上菜单栏里的“共享目录”按钮，就会弹出一个目录选择对话框。用户只需选择要共享的目录单击“打开”即可，如图 12.41 所示。

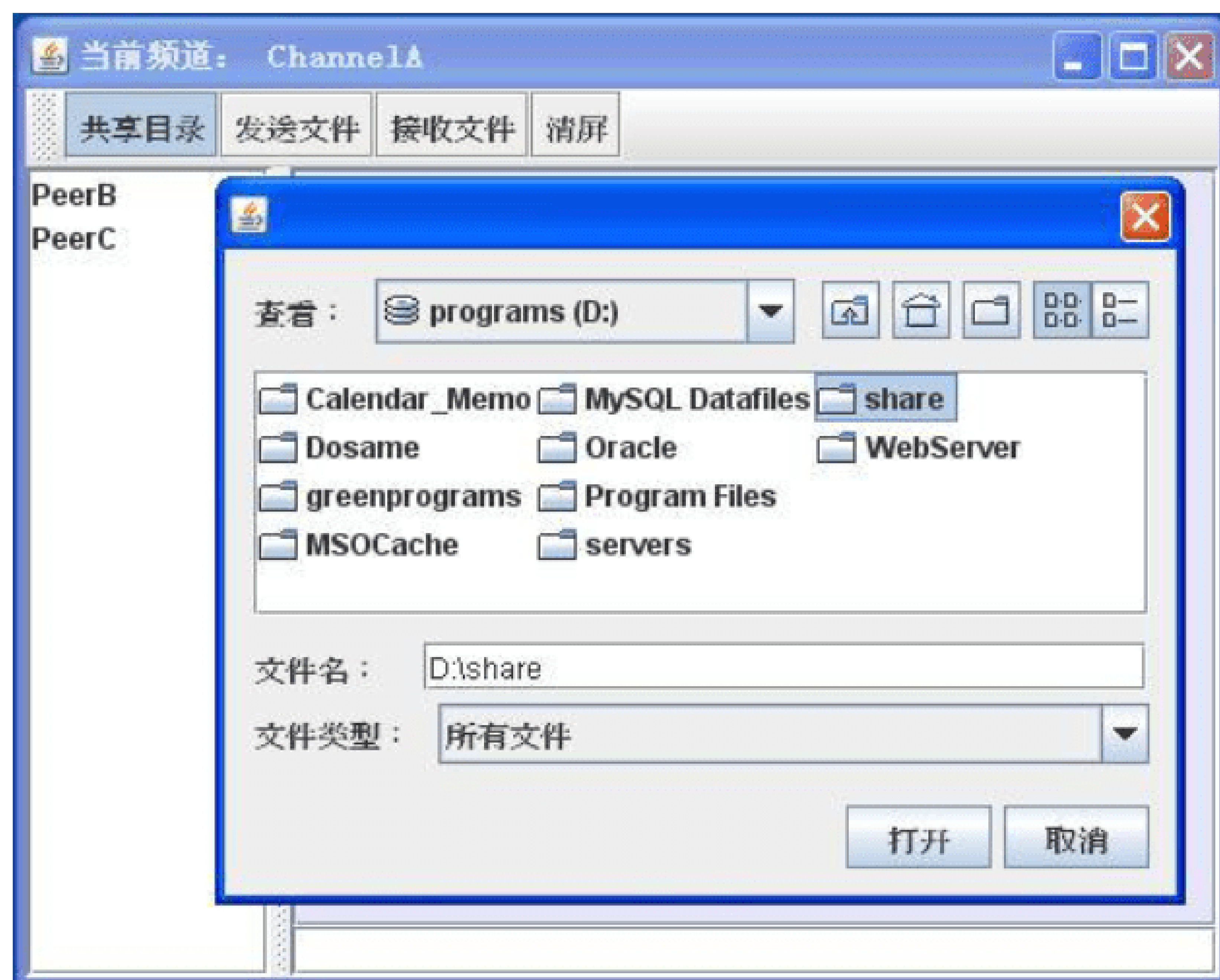


图 12.41　Peer 结点共享本地目录时的操作示意图

通过如图 12.41 所示的操作，就可以把 PeerB 主机上 D 盘下的 share 目录共享出去了，系统会发出相应的通知消息。也可以选择 PeerB，右击，在弹出的快捷菜单中选择“显示共享”选项来查看详细的共享列表和文件内容，如图 12.42 所示。

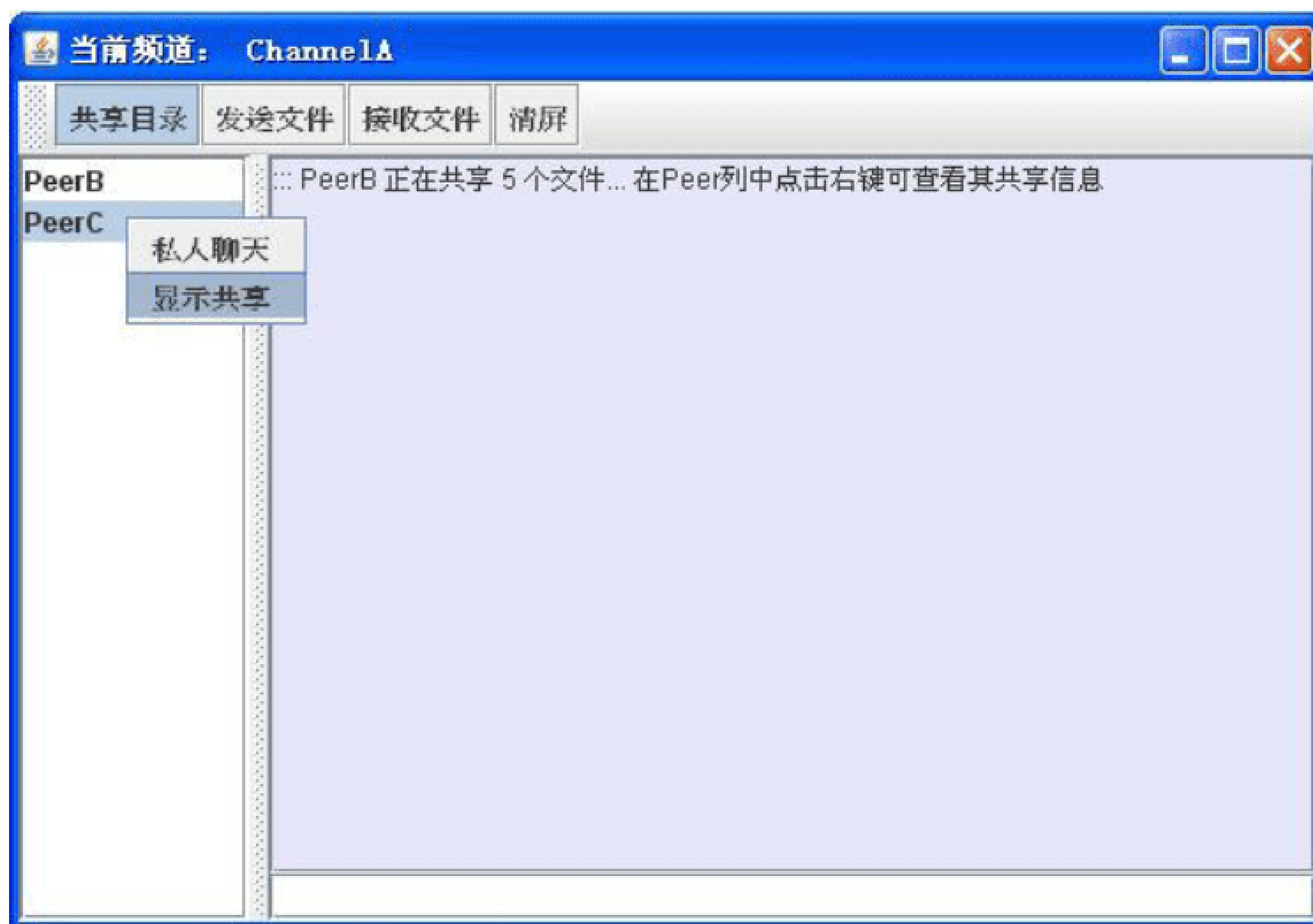


图 12.42　显示 Peer 共享的功能

PeerB 将文件共享出去后，与 PeerB 同频道的 PeerC 结点也会收到系统的通知消息，那么 PeerC 就可以操作 PeerB 的共享了。如图 12.43 所示，是在 PeerC 中查看 PeerB 的共享文件列表和文件内容。

在图 12.43 所示的界面中，左侧显示的是共享文件的列表，右侧显示的是文件的内容。

也可以右击共享列表中的某一文件，选择“另存为”将其存储到本地，如图 12.44 所示。

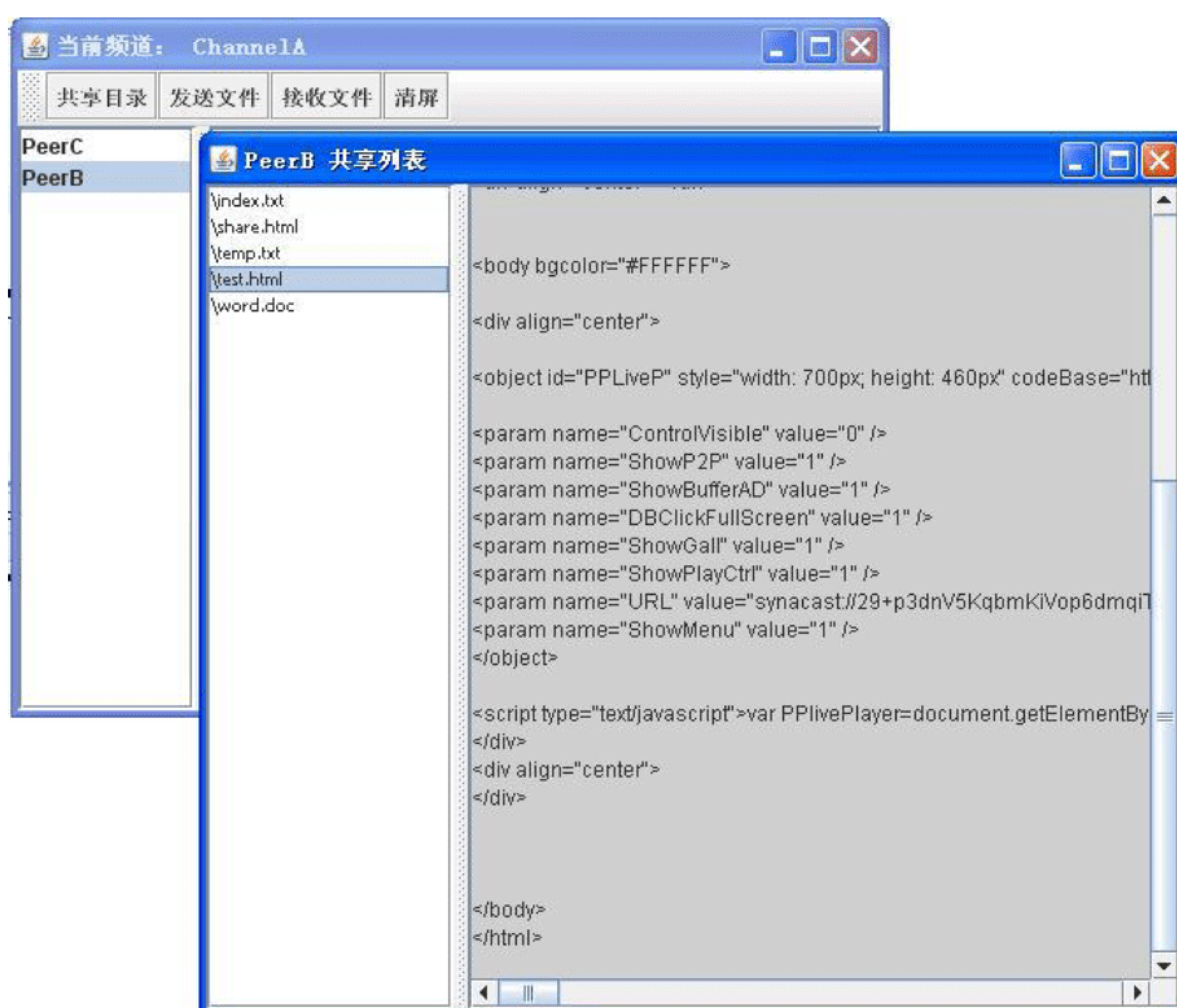


图 12.43　查看 Peer 共享内容的操作

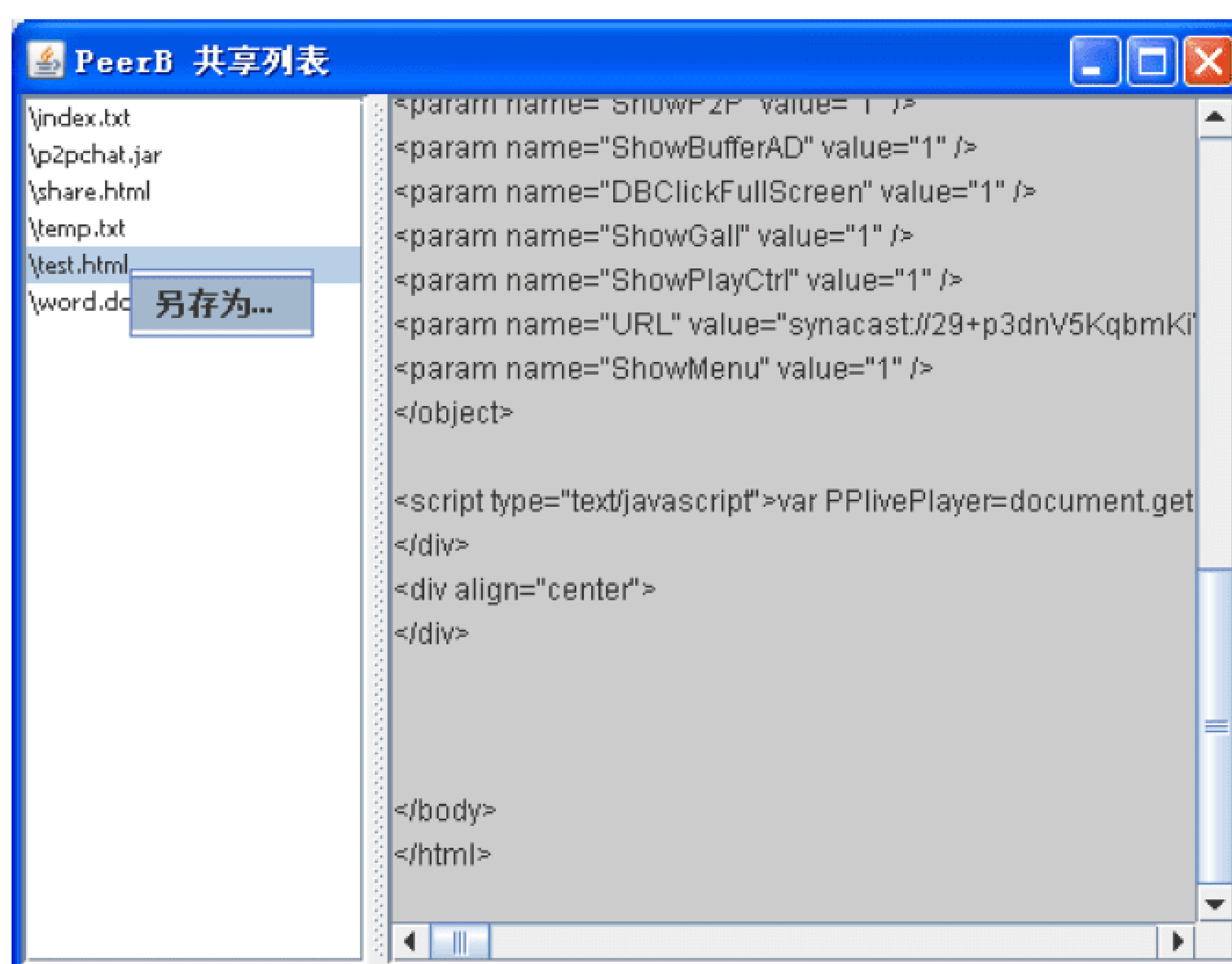


图 12.44　将共享文件存储到本地的操作

在文件共享上本系统的设计目标是：在同一个 P2P 频道内，任何 Peer 都可以共享自己本地的目录，其他的所有 Peer 都可以分享这个目录的内容，包括查看文件列表、显示文件内容、下载共享文件等。关于文件共享中其他的特征功能，读者可以自己进行其他各方面的测试。

12.6.3　文件的发送与接收

文件的发送与接收，主要是用来实现 Peer 结点间交互文件的功能，Peer 结点可以向其他在同一频道内的所有结点发送文件。文件的发送过程是一个广播过程，每个结点都可以接收到发送的文件。

1．文件的发送

在频道主界面中，单击菜单栏的“发送文件”按钮，这时会弹出一个文件选择对话框。选择要发送的文件，单击“打开”按钮就可以了，此时，选择的文件就会发送到所有的 Peer 中，如图 12.45 所示。这是在 PeerB 主机上的截图，这样 PeerB 就把一个名为 promises.doc 的文件发送给所有的结点了，包括自身。发送完成后，在消息窗口中也会打印出通知消息，如图 12.46 所示。

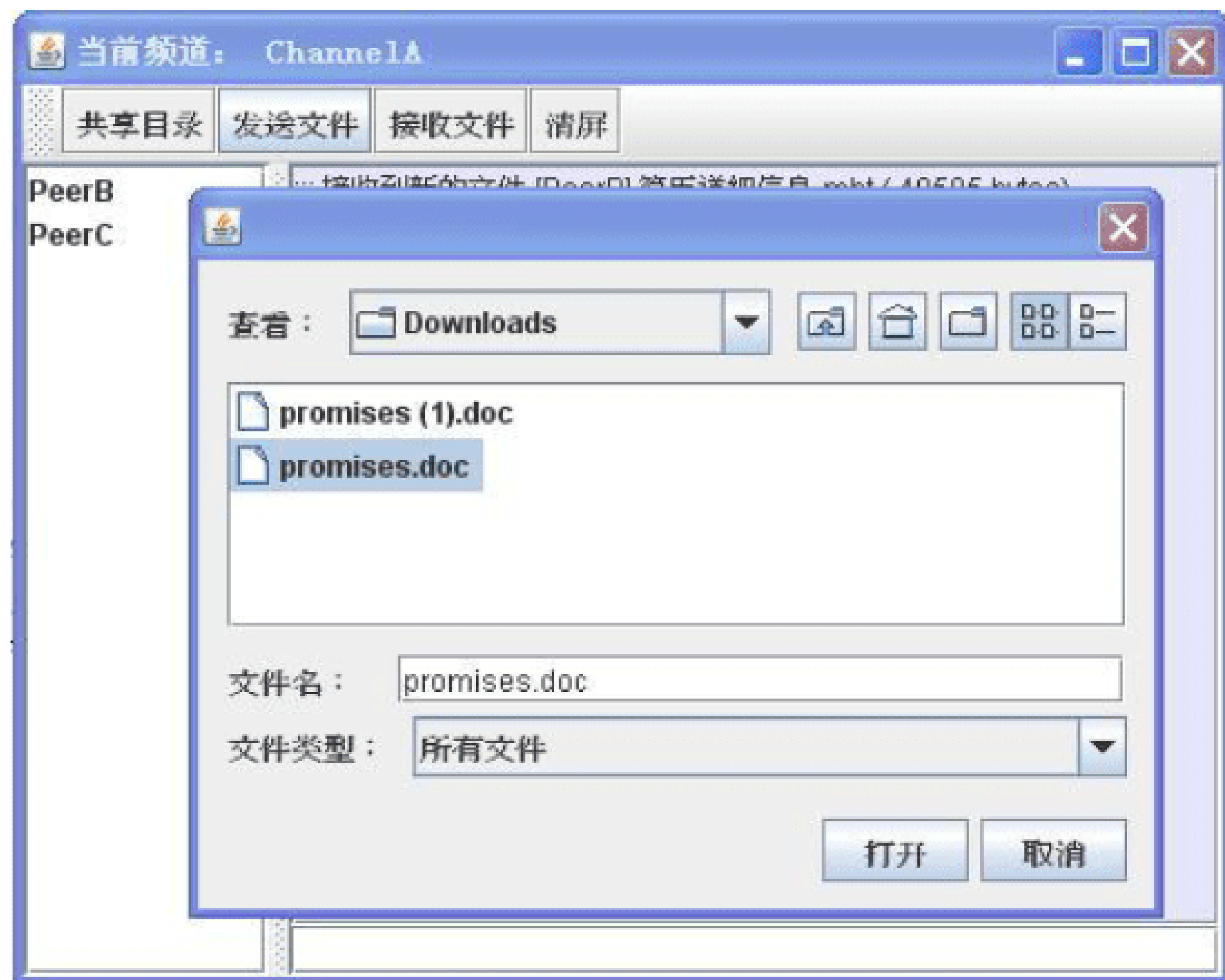


图 12.45　发送文件的操作示意图



图 12.46　当有文件发送的时候系统打印通知消息的情况

在图 12.46 中显示的是系统的通知消息，其中最下面一条就是刚才发送的文件消息，系统消息的通知为：“接收到新的文件[PeerB] promises.doc (25088 bytes)”，指的就是接收到一个来自 PeerB 的、名为 promises.doc、大小为 25088 bytes 的文件。

注意：本系统在开发的时候设定了发送文件的大小，文件小于 65000 B 的文件才允许被发送出去，在源代码中读者可以自己改动这种设定。

2．文件的接收

文件的接收也很简单，当收到系统的通知消息后，证明所有的 Peer 都接收到了这样一个文件，这时只需单击“接收文件”按钮就会弹出一个“接收文件界面”。在这个界面中就可以查看接收到的文件了，如图 12.47 所示。

图 12.47 中所示的是在 PeerC 主机上查看“接收文件”时的截图，可以看到，在本系统中先后发送了 3 个文件，在接收文件界面中显示的内容与系统通知消息的内容是完全一

致的。这就说明，Peer 结点发送的文件都能被接收到。



图 12.47　文件接收的操作界面

12.6.4　文件的查看和下载

在“接收到的文件”界面中，有两个区域，一个是用于显示文件基本信息的，如“[PeerB] promises.doc (25088 bytes)”，这个信息指的是 promises.doc 文件是由 PeerB 发送的，大小为 25088 B，另一个区域用来显示文件的内容，当双击上方区域的文件基本信息后，下方区域中就会显示相应的文件内容，如图 12.48 所示。这是双击“[PeerB] 简历详细信息.mht (48505 B)”文件时，显示的文件内容，这个界面大小是可以自由拉伸的。

文件下载的方法也很简单，只需选中要下载的文件，然后右击，在弹出的快捷菜单中选择“另存为”选项即可。这时会弹出文件对话框，选择一个存储路径，就可以将接收到的文件存储到本地了，如图 12.49 所示。

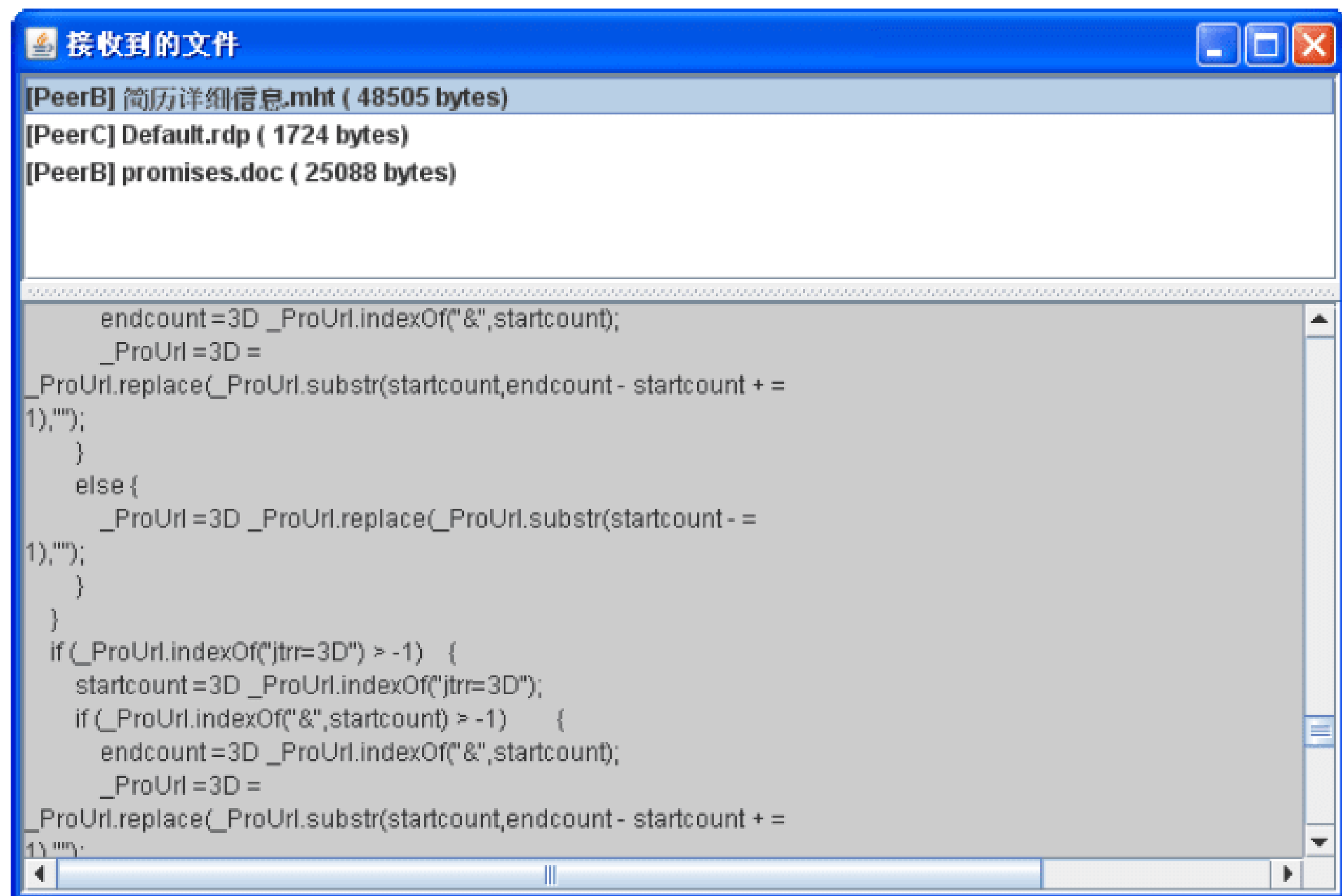


图 12.48　显示接收到的文件内容的界面

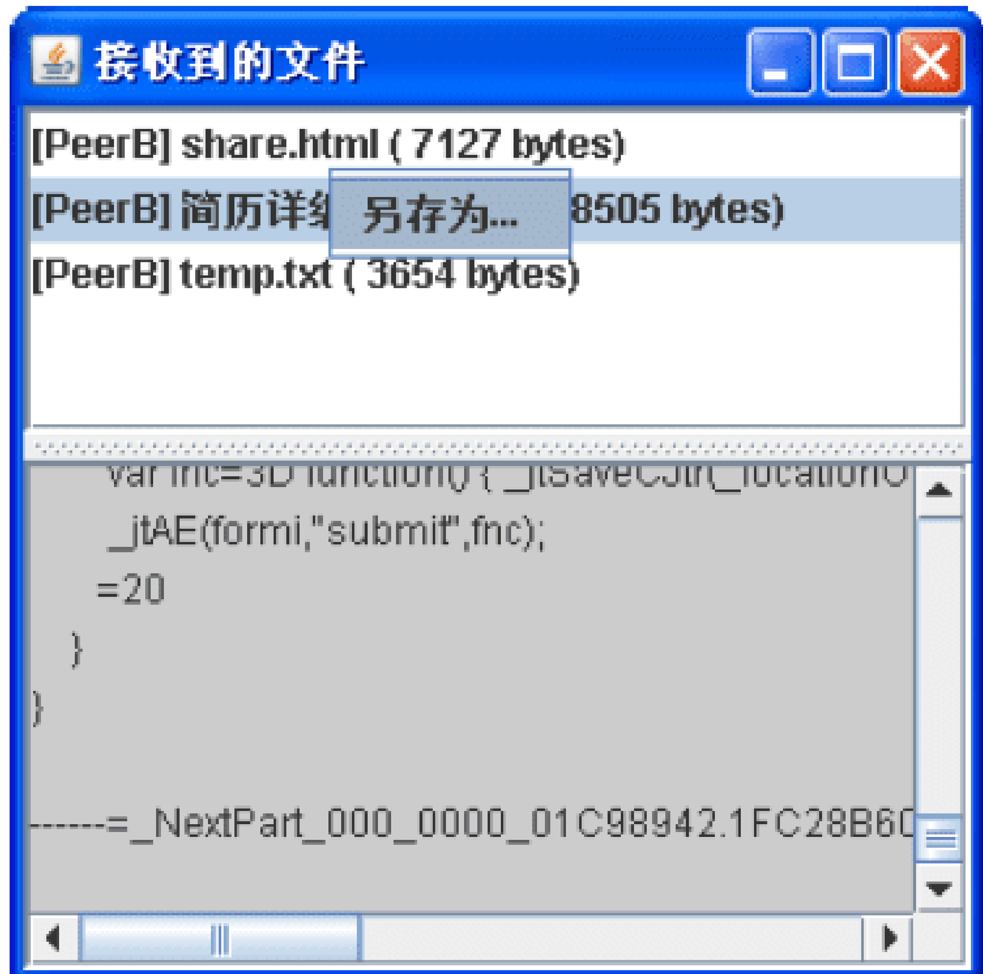


图 12.49　将接收到的文件存储到本地的操作

图 12.49 所示的是将接收到的文件存储到本地的过程，在接收文件界面的接收文件列

表中，选择文件名右击，在弹出的快捷菜单中选择“另存为”操作就可以了。

12.6.5　点对点的私人会话

点对点的私人会话，指的是两个 Peer 间的相互通信，它的会话是建立在两个 Peer 之间的，会话过程只有参与会话的两个 Peer 完成，会话内容对其他 Peer 不可见。

在频道主界面中，从 Peer 列表中选择一个你想与之会话的 Peer 结点，右击，在弹出的快捷菜单中选择“私人聊天”选项，这样就可以与你选择的 Peer 间建立一个私人的会话通道，如图 12.50 所示。

图 12.50 所示的是在 PeerB 主机上、由 PeerB 主动发起的一个与 PeerC 的私人会话，这个会话建立完成后，就会弹出一个私有的、连接到 PeerC 的会话框，如图 12.51 所示。

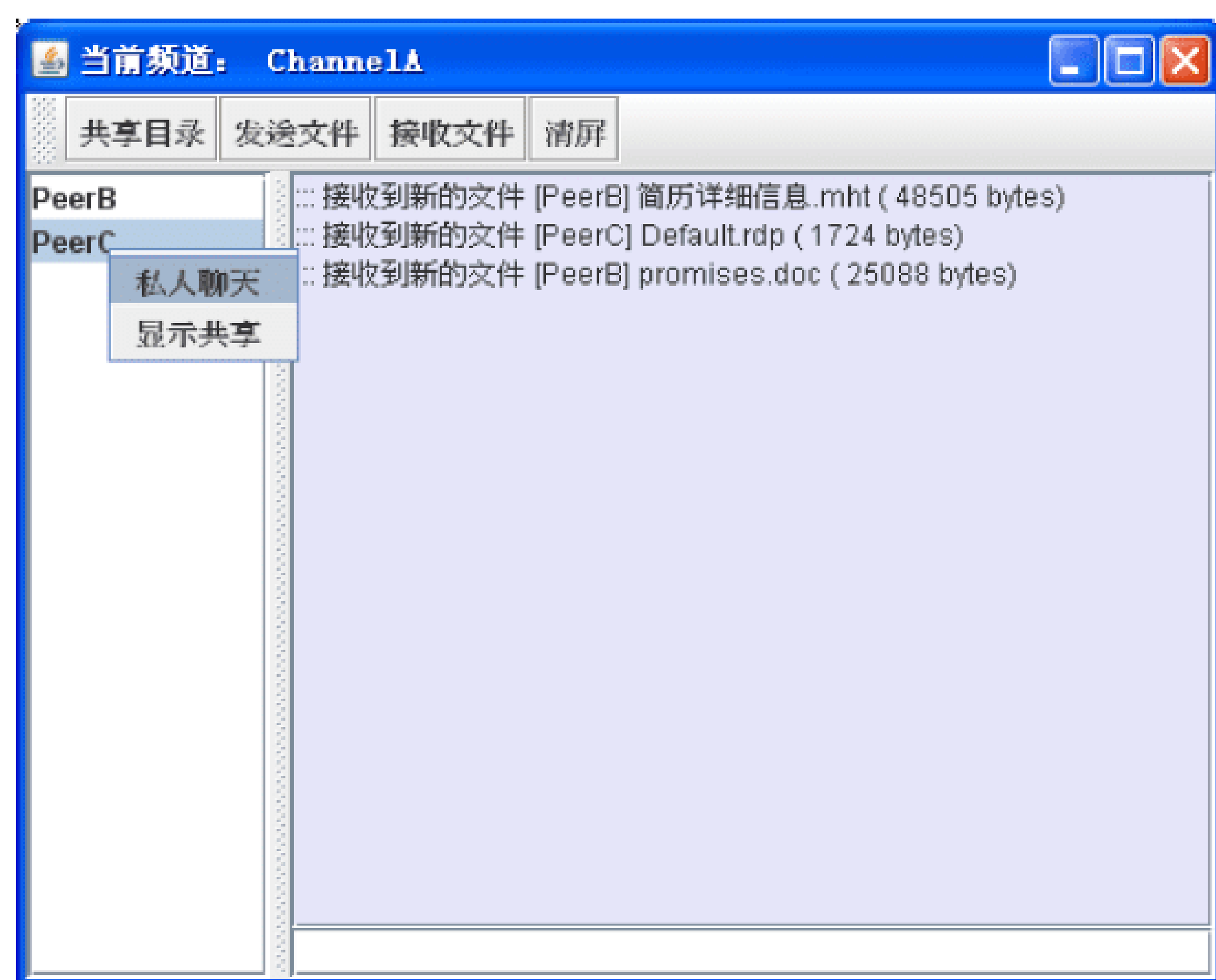


图 12.50　Peer 间私人会话的创建示意图

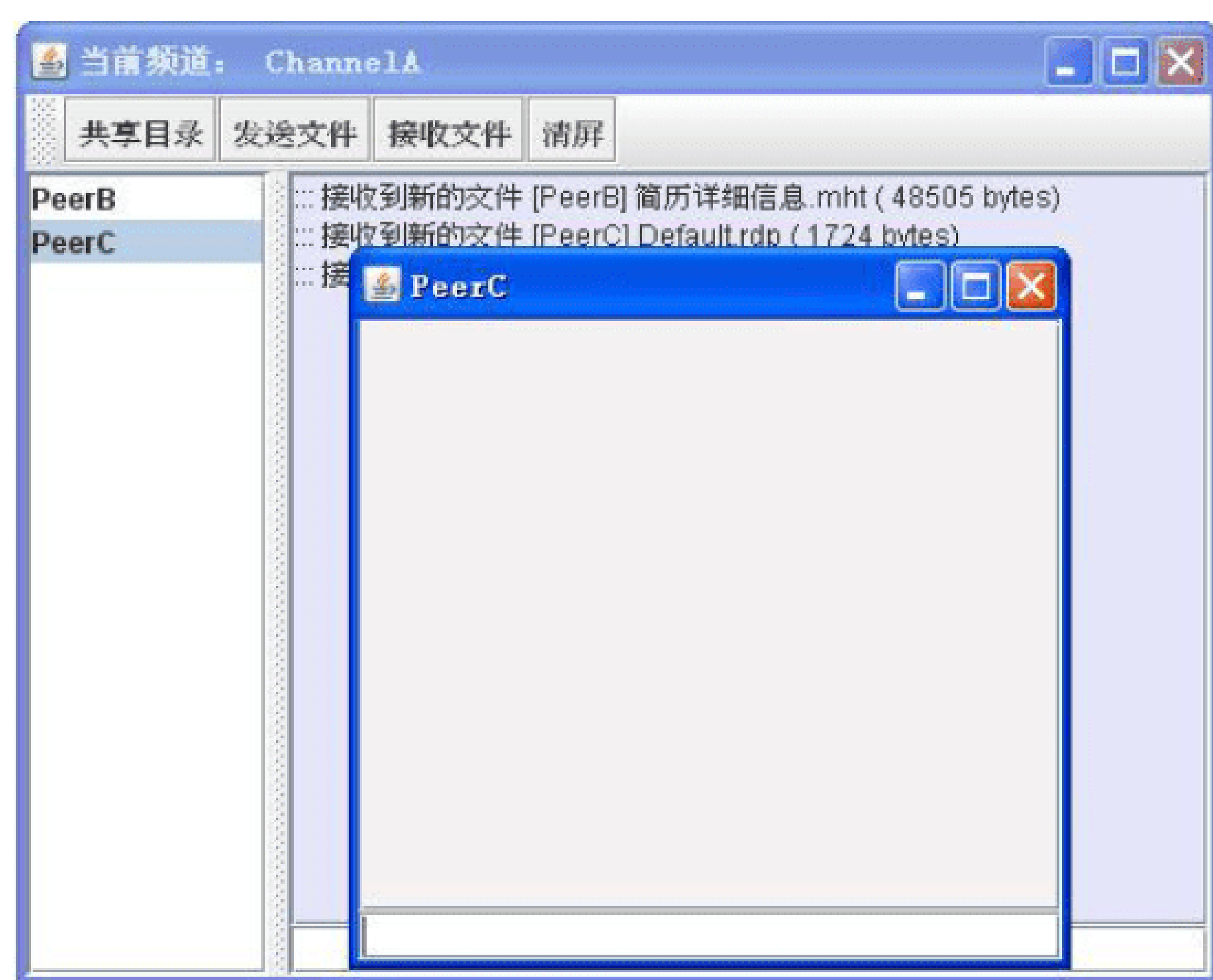


图 12.51　PeerB 创建的到 PeerC 的私人会话

由图 12.51 可以清楚地看到，这个会话框的标题为 PeerC，也就是这是一个指向 PeerC 的会话，在这个会话框中出现的内容只有 PeerC 能看到。此时，可以向 PeerC 发送一条消息，如图 12.52 所示。会话的内容会显示到消息框中，而且还会清楚地标示出这句话是由谁说的。图 12.52 中显示了 PeerB 连续发出了两条消息，此时在 PeerC 的一端就会自动弹出这个会话框，单独接收到 PeerB 发送过来的消息，如图 12.53 所示。



图 12.52　在私人会话框中发送消息的过程

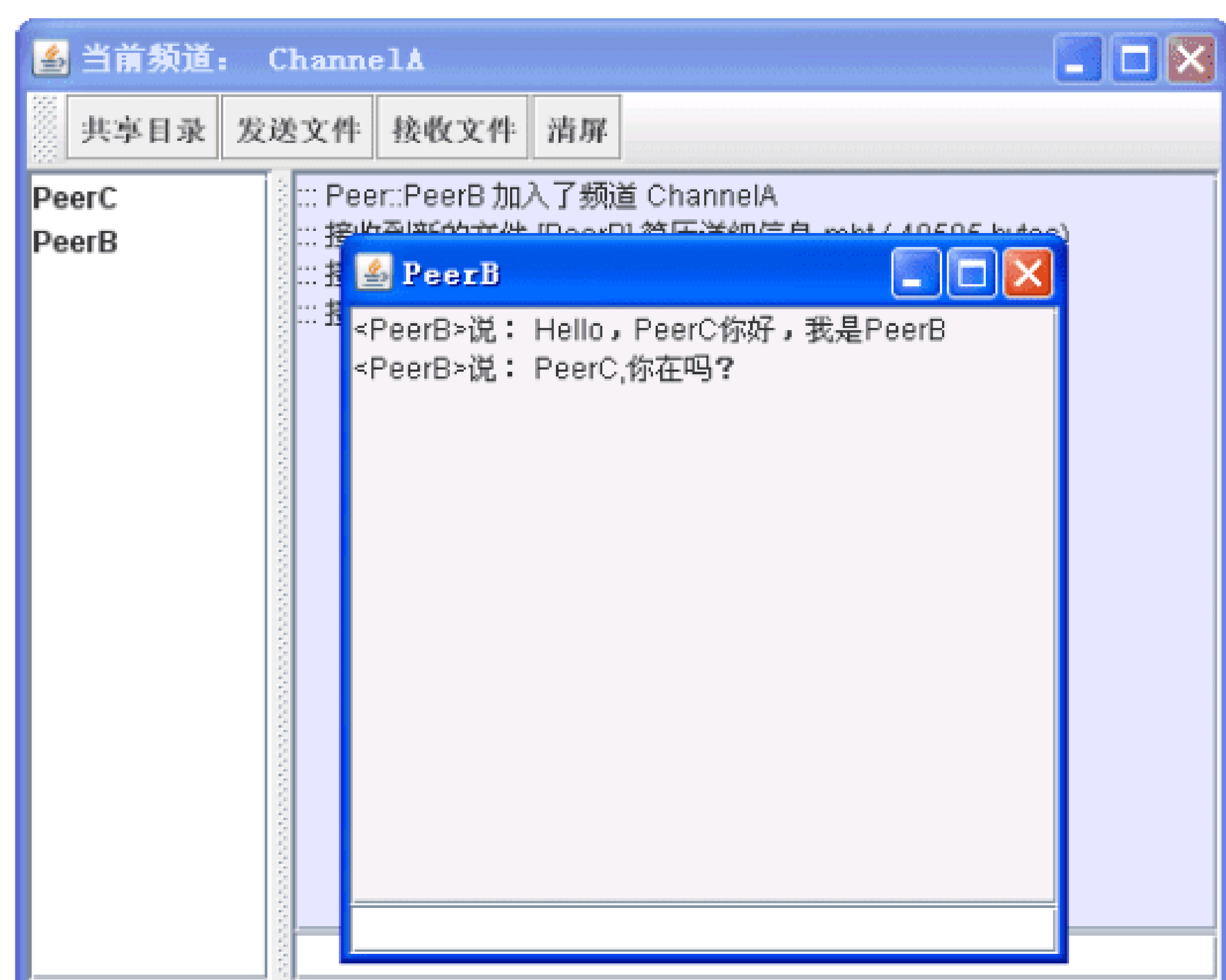


图 12.53　PeerC 一端接收到的会话消息

由图 12.53 中看出，PeerC 一端接收到了来自 PeerB 的会话，内容也与 PeerB 发送的内容完全一致，这样，两个 Peer 之间就可以进行点对点的私人会话了。图 12.54 所示的就是 PeerB 与 PeerC 之间消息交互的对比效果。



图 12.54　两个 Peer 间消息对等交互的效果图

这样，就可以实现 Peer 间的私人会话功能了，在会话过程中还可以输入“/cls”或“/clear”来执行清屏操作。

以上的测试验证了系统的一些主要功能。从测试的效果来看，基于 P2P 的即时通信系统，达到了预先的设计目标，能够实现基本的 P2P 机制，可以完成即时通信系统中文件共享、数据交互、私人会话等基本功能，实现了基本需求。由此证明系统的开发思路是正确的，开发结果也达到的预期的效果。

当然，本节所描述的测试都是基于基本应用的测试，从工程应用的角度而言，还需要进行一系列用例测试、压力测试、容错测试等，对这些有兴趣的读者可以自己进行相关测试。

12.7　程序的打包与发布

从一个 Java 应用系统的完整开发过程而言，当所有的开发工作完成以后，还需要对程序进行打包并发布。对于一个使用者而言，他并不关心程序实现的细节，也不关心哪个类是主类、哪个方法是入口方法等，使用者所追求的就是使用简单方便，程序拿过来就可以用、双击就可以运行。要实现这个目的，就需要对开发的源程序进行打包，生成可直接运行 Jar 文件或 exe 文件。

12.7.1　Java 源程序打包的几个概念

要将 Java 源代码编译后的字节码程序（.class 文件）变成一个在主机上可直接运行的程序包，有两种方法可以实现，一种是将其打包成 Jar 文件，另一种就是打包成可执行的 exe 文件。要生成这两种类型的文件，有几个概念需要先了解一下，后文会用到。

1．什么是Jar文件

Jar 文件实际上是 class 文件的 ZIP 压缩存档，它本身是将许多信息经过封装后形成的捆绑体，是一个压缩文件。有很多工具可以操作这种格式的文件，通过 WINRAR 类的程序就可以将其解压缩后打开，通过 Java(TM) Platform SE binary 工具可以让其直接运行，也可以通过一个.bat 的批处理文件让其在命令行中运行等，基本来说，能将 Java Code 打包成 Jar 文件，也算是一个可执行的文件了。

注意：这里所说的 Java Code 文件，指的是 Java 源码编译后的字节码文件。

2．Manifest文件

上文已经说了，Jar 文件是一个压缩文件，正是因为这个原因，Jar 文件本身并不能表达所包含应用程序的标签信息，此时就需要用到一个名叫 Manifest 的文件了。

为了要提供存档的标签信息，Jar 文件指定了一个特定目录来存放标签信息：META-INF 目录，在此目录中的 MANIFEST.MF 文件，它就是 Jar 的 manifest 文件，包含了 Jar 文件的内容描述，并在运行时向 Jvm 提供应用程序的信息，大多数 Jar 文件含有一个默认生成的 Manifest 文件，执行 Jar 命令或使用 zip 工具，都可以产生它。

如果是由 Jar 命令产生的 manifest 文件，其文件形式如下。

- Manifest-Version: 1.0；
- Created-By:1.6.0-beta；
- (Sun Microsystems Inc.)。

这些信息没什么用，仅仅告诉我们使用的是 1.0 的 manifest 文件，第 1 行定义 manifest 的格式，第 2 行说明使用 SUN 的 JDK 1.6 的 Jar 工具生成该文件。如果 manifest 文件是由其他（如 ant）创建的，则将会出现“Created-By: Ant 1.2”之类的内容，如果是自己创建 Manifest 文件，则可以加入自己的一些相关信息。

3．Manifest文件与Jar包的运行

现在来体验一下 Manifest 文件的作用，如果现在将本系统的 Java 应用程序打包在一个名为 p2pchat 的 Jar 包中，此包中 main class 为 p2p.chat.Main，那么可以用以下的命令来运行：

```
java -classpath p2pchat.jar  p2p.chat.Mani
```

从用户的角度来说，这样来运行一个程序显然是太麻烦了，需要输入一长串代码说明，还要知道系统中 main class 的位置，这种方法肯定是不可取的。现在来创建自己的 Manifest 文件，如下：

- Manifest-Version: 1.0；
- Created-By: gl p2pchat case；
- Main-Class: p2p.chat.Main。

这样我们就可以使用如下命令来运行程序了。

```
java -jar p2pchat.jar
```

这种方式运行就明显简单多了，如果将 Jar 文件的属性关联至 Java(TM) Platform SE binary 上，直接双击就可以运行了。

12.7.2　生成 Jar 文件的基本方法

要将 Java Code 文件打包成 Jar 文件，有多种方法，JDK 开发工具包中提供有相应的命令，可以直接生成 Jar 文件；一些开发工具如 Eclipse、Netbeans 等，也都提供了简易的生成 Jar 文件的方法。下面就简要地讲一下生成 Jar 文件的基本方法。

1．通过jar命令生成Jar文件

在 JDK 开发工具包中提供有专门用于生成 Jar 文件的命令，就是 jar 命令。在安装有 JDK 并配置好环境变量的主机上，进入命令行后，直接输入 jar –help 命令，就可以查看关于 jar 命令的详细信息。通过 jar 命令生成 Jar 包的完整命令格式是：

```
jar {ctxu}[vfm0Mi] [jar-文件] [manifest-文件] [-C 目录] 文件名 ...
```

命令中的相关选项：

- -c：创建新的存档；
- -t：列出存档内容的列表；
- -x：展开存档中的命名的（或所有的）文件；
- -u：更新已存在的存档；
- -v：生成详细输出到标准输出上；
- -f：指定存档文件名；
- -m：包含来自标明文件的标明信息；
- -0：只存储方式；未用 ZIP 压缩格式；
- -M：不产生所有项的清单（manifest）文件；
- -i：为指定的 jar 文件产生索引信息；
- -C：改变到指定的目录，并且包含下列文件。

如果一个文件名是一个目录，它将被递归处理，找到这个目录及其子目录下的每一个文件。清单（manifest）文件名和存档文件名都需要被指定，按 m 和 f 标志指定的相同顺序。

按照以上的命令方法，要将两个 class 文件存档到一个名为 classes.jar 的存档文件中，需要执行以下命令：

```
jar cvf classes.jar Java1.class Java2.class
```

也可以将与清单文件（就是 manifest 文件，上文已有讲解）结合起来使用。例如，用一个存在的清单（manifest）文件 mymanifest 将 dir/目录下的所有文件存档到一个名为 classes.jar 的存档文件中，需要执行以下命令：

```
jar cvfm classes.jar mymanifest -C dir/.
```

以上就是用 JDK 本身提供的 jar 命令来生成 Jar 文件的过程和基本方法，这种用命令的方法适合一些初学者打包一些简单的、数量少的 class 文件，而一些系统性的工程打包，

一般都借助一些工具平台来完成。

2．在Eclipse中生成Jar文件的基本方法

在 Eclipse 中，对普通类导出 Jar 包很简单，只需执行以下几个操作步骤就可以了。

注意：这里所说的普通类，指此类包含 main()方法，并且没有用到别的 Jar 包。

（1）在 Eclipse 中右击要导出的类或者工程名，在弹出的快捷菜单中选择 Export 子选项。

（2）在弹出的对话框中，选择 Java 文件 | JAR file，单击 next 按钮；

（3）在 JAR file 后面的文本框中选择要生成的 Jar 包的位置以及名字，注意在 Export generated class files and resources 和 Export java source files and resources 前面打上勾，单击 next 按钮。

（4）单击两次 next 按钮，到达 JAR Manifest Specification。注意在最底下的 Main class 后面的文本框中选择 Jar 包的入口类。单击 Finish 按钮完成。

以上就可以生成 Jar 包文件了，可以在 dos 环境下，进入 Jar 所在的目录，运行 java -jar 名字.jar，检测运行是否正确。

3．需要引入外部Jar包时的打包方法

以上所讲的方法中，都是假定工程中没有引用其他 Jar 的情况下进行的。在实际开发中，一个 Java 工程很可能引用了多个外部的 Jar 包，这一类的工程在打包的时候，还需要进行额外的处理。

比如，在开发某一个工程中需要进行连接 Oracle 数据库的操作，此工程中引用数据库的驱动包 oracl.jar，在打包的时候会连同 oracle.jar 一并打包进去，操作方法如下。

（1）先把要导出的类按照上面在 Eclipse 中生成 Jar 文件的基本方法所描述的步骤导出形成 jar 包，比如叫 first.jar。

（2）新建一个文件夹 main，比如在 D 盘根目录下。

（3）把 main.jar 和 oracl.jar 复制到 main 文件下，右击 first.jar，解压到当前文件夹。把 META-INF\MANIFEST.MF 剪切到另外一个地方 （比如是桌面）。

（4）右击 oracl.jar，解压到当前文件夹。

（5）在 dos 环境下，进入到 D 盘的 main 文件夹下，执行 jar cvfm new.jar meta-inf/manifest.mf .，不要忘了最后面的点。

（6）用压缩工具打开新生成的 new.jar，用你放在桌面的 META-INF\MANIFEST.MF 覆盖 new.jar 原有的即可。

这是一个很笨的用 jar 命令进行引入外部 Jar 包的打包操作过程，如果借助其他的一些打包工具如 Fatjar 等，可以很方便地将外部引用包一起生成一个统一的 Jar 包文件。

12.7.3　Fatjar 打包插件的用法

Fat Jar Eclipse Plug-In 是一个可以将 Eclipse Java Project 的所有资源打包进一个可执行 Jar 文件的小工具，可以方便地完成各种打包任务。

用过 Eclipse 的读者应该都知道，其自带的生成 Jar 包的功能不够强大。而 Fat Jar 是 Eclipse 的一个插件，特别是 Fat Jar 可以打包成可执行 Jar 包，并且在图片等其他资源、引用外包方面使用起来非常方便，所以，这里就简要讲一下这个插件的使用方法。

1．Fatjar的安装

FatJar 在 Eclipse 中的安装有两种方法，一种是通过 Eclipse 在线更新方法来安装，另一种是将其下载以插件的方式来安装。

（1）在线更新安装

通过 Eclipse 的在线更新方法可以直接将 Fatjar 插件安装到 Eclispe 中，具体方法是在 Eclipse 工具栏中，选择 help | software updates | Available Software | Add site 命令。在弹出的界面中，填写 site 的 URL 地址 http://kurucz-grafika.de/fatjar，这个地址就是 FatJar 插件的更新站点，单击 OK 按钮后，就会出现 FatJar 的安装信息，如图 12.55 所示。

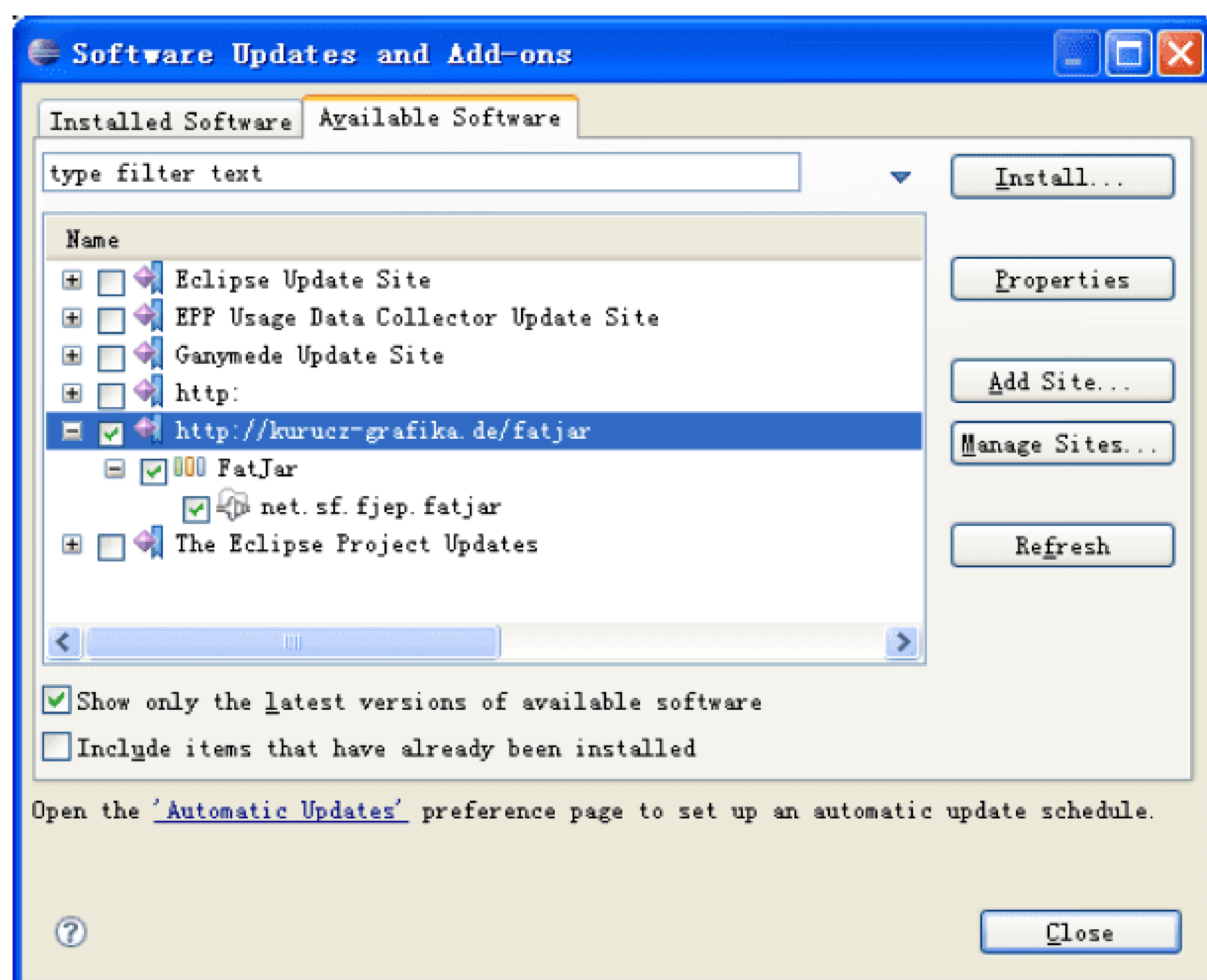


图 12.55　Fatjar 插件的在线安装信息

如图 12.55 所示的界面中，选中此站点中的 Fatjar 文件，直接单击 Install 按钮就可以进行在线安装 Fatjar 插件了。

（2）Eclipse 插件安装方法

通过互联网可以免费下载 Fatjar 的安装包，具体下载地址，可以通过搜索引擎轻易地找到，在本书的随书光盘中也会提供 Fatjar 软件包。

下载完 Fatjar 软件包，解压后，将..\net.sf.fjep.fatjar_0.0.25\plugins 中的内容复制到 Eclipse 安装路径\eclipse\plugins 的目录下，然后重启 Eclipse，避免 Fat Jar 被认不出来，在 Eclipse 启动时使用-clean 参数。

2．Fatjar的使用

安装完 Fatjar 插件后就可以使用了，使用方法也很简单，选择要打包的项目，直接右击，在弹出的快捷菜单中选择 Build Fat Jar 选项即可，如图 12.56 所示。

按照图 12.56 所示的操作后，接着就会弹出 Fat Jar 操作的界面，按照界面提示和说明，直接操作即可，在下文会讲到将本系统用 FatJar 插件打包的具体方法。

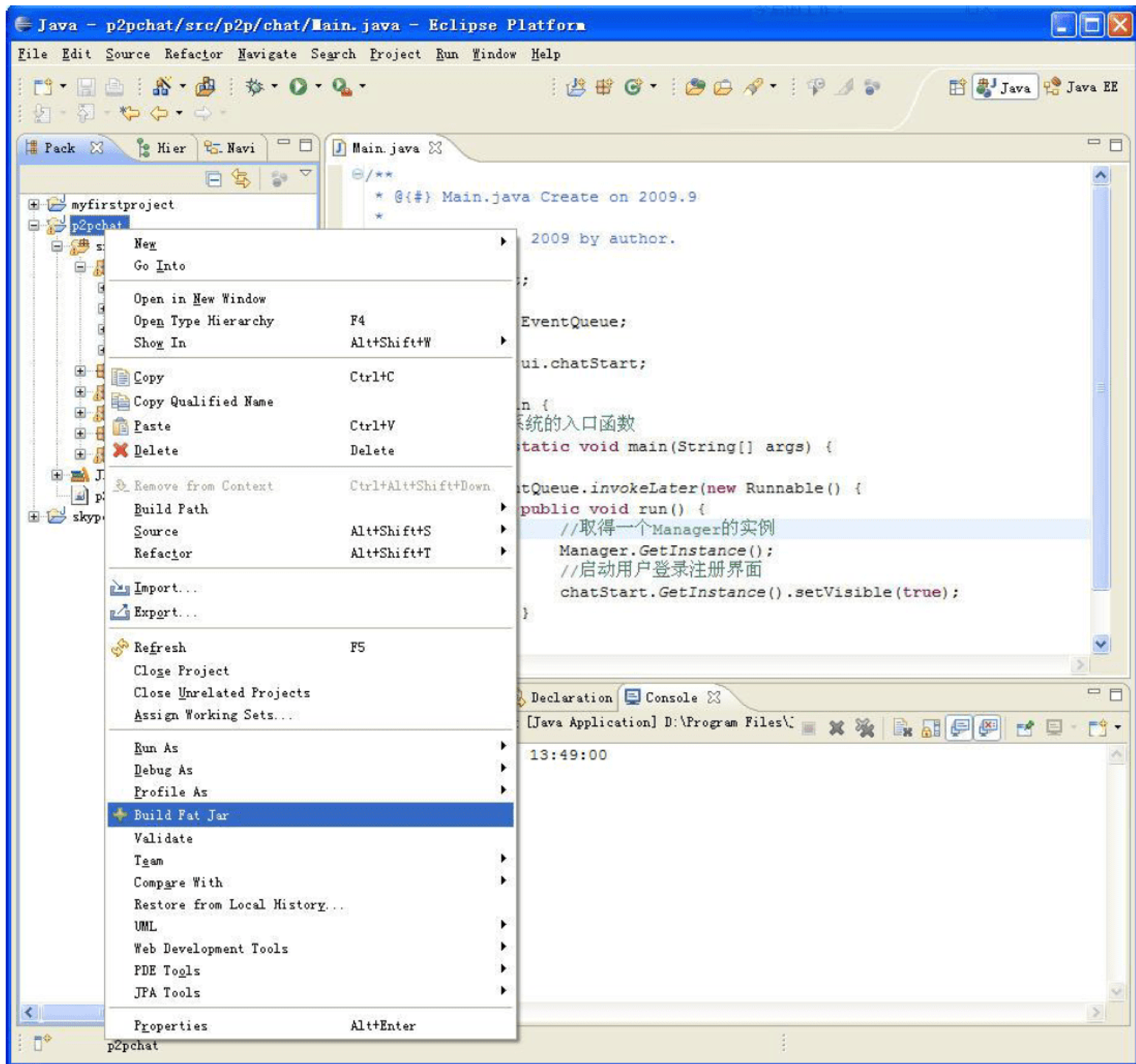


图 12.56　FatJar 插件使用界面

12.7.4　将系统打包成 Jar 文件发布

本系统在开发过程中无须用到第三方 Jar 包，所以直接用 Eclipse 就可以打包成 Jar 文件。很简单，上文讲了 Fat Jar 插件的使用，这里就演示一下如何用 Fat Jar 将本系统打包成可直接双击运行的 Jar 文件。

在安装完 Fat Jar 的 Eclipse 中，选中 p2pchat 工程文件，右击，在弹出的快捷菜单中选择 Build Fat Jar，就会弹出一个向导对话框界面，如图 12.57 所示。Jar-Name 项，就是设置生成 Jar 文件的名字，直接勾选 use extern Jar-Name 会自动将此 Jar 文件生成到工程目录下，也可以选择设置到任意位置。Manifest 项，如无特殊说明，无须再进行设置，系统会自动生成，Main-Class 项，直接定位到主类 p2p.chat.Main 即可。其他选项的意思读者自己去了解，就本系统而言，选择默认设置就可了。单击 Next 按钮，进入如图 12.58 所示的界面。

图 12.58 所示的界面，主要是引入外部的 Jar，在需要引入第三方 Jar 的工程中此步的设置非常重要。本系统中没用到任何外部的依赖 Jar，不需进行任何设置，直接单击 Finish

按钮即可。这样，本系统的 Jar 文件包就生成了。生成的效果如图 12.59 所示。

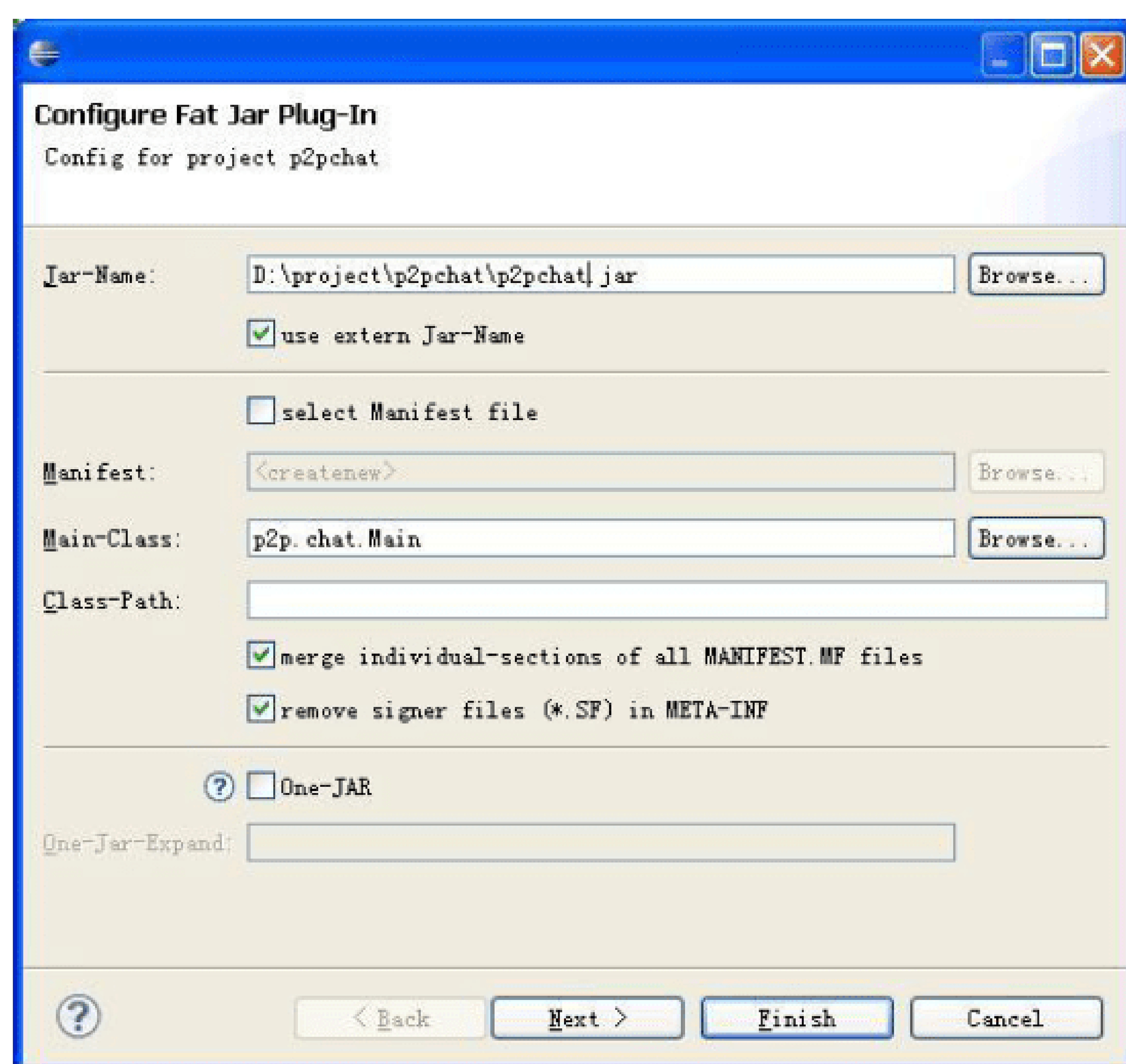


图 12.57　使用 Fat jar 打包时的选项设置界面

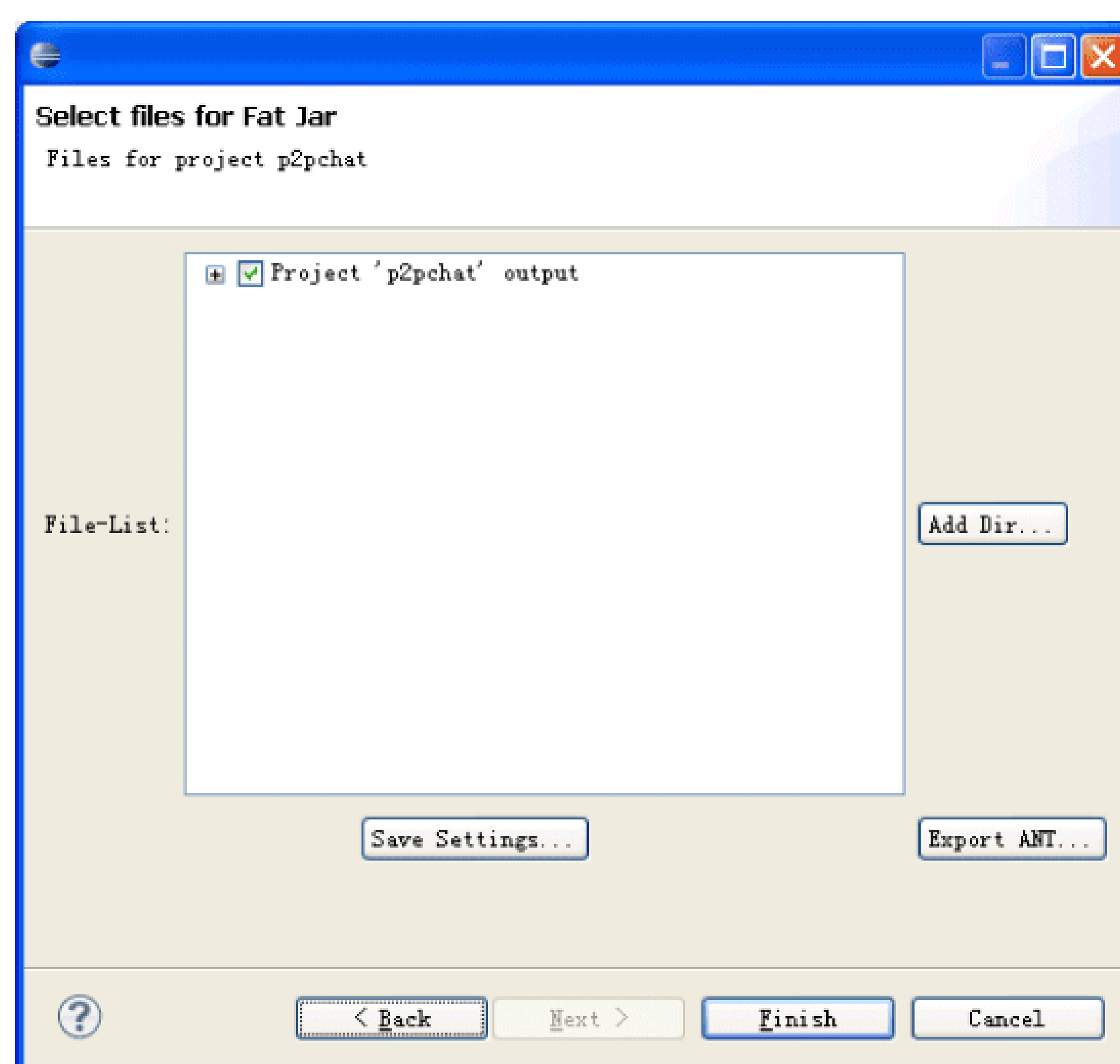


图 12.58　Fatjar 中引入外部 Jar 包的操作界面

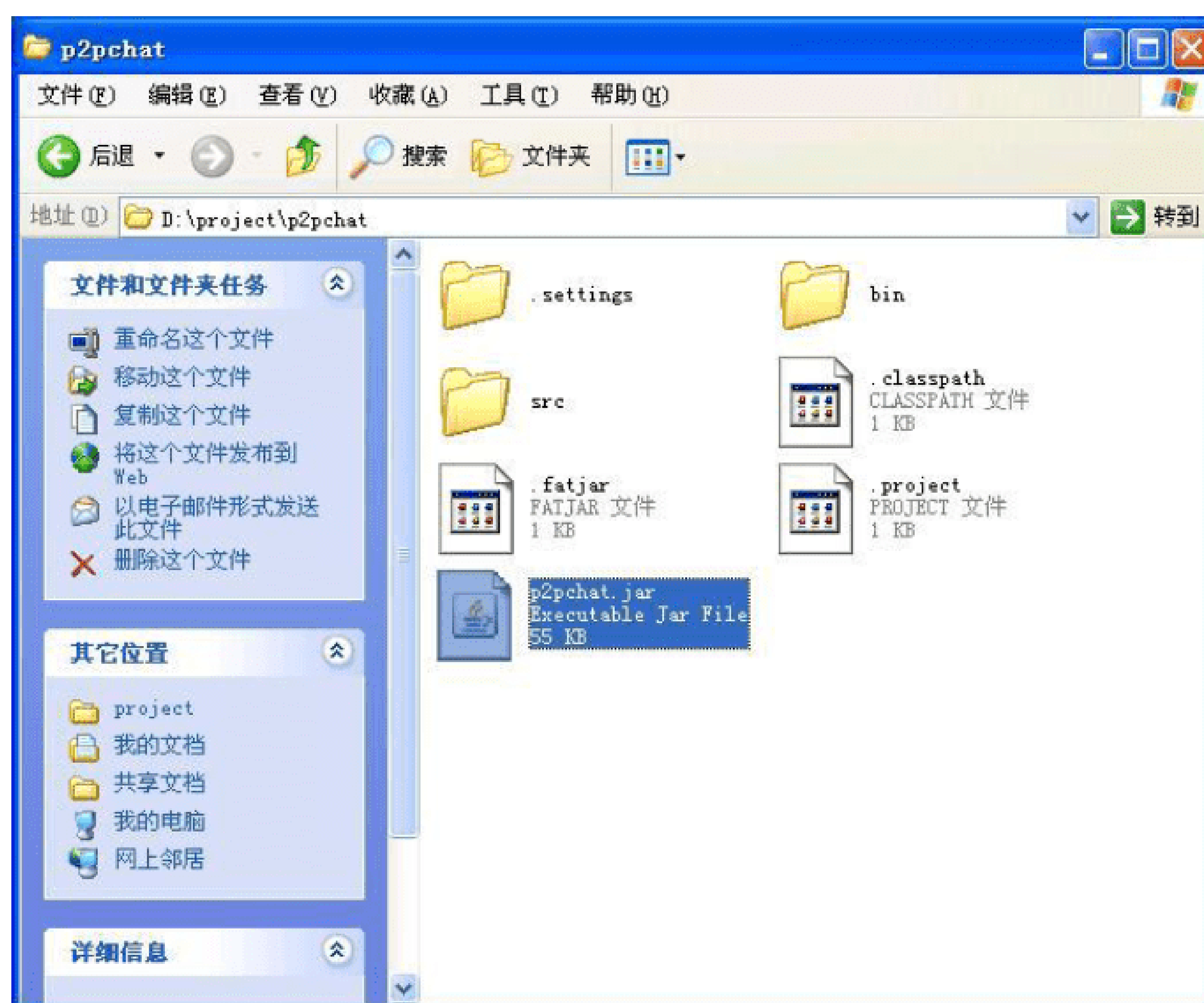


图 12.59　生成的 Jar 文件后示意图

这样，一个 p2pchat 工程的 Jar 文件包就生成了。双击生成的 p2pchat.jar 或者在命令行下输入如下命令：

```
java -jar p2pchat.jar
```

就可以启动打包后的 P2P 即时系统运行。

12.7.5　生成 EXE 文件发布

Jar 文件需要在装有 Jdk 的环境中才能运行，有时还需要在命令行下执行，这给程序的

运行带来了不便通过一些第三方工具，还可以将 Jar 文件直接生成双击即可运行的 exe 文件，常用的工具有 jar2Exe、exe4j 等。下面就讲一下如何运用 jar2Exe 工具将 p2pchat.jar 生成 exe 文件。

jar2Exe 是一款收费软件，可以将 Jar 文件转化为 Exe 可执行文件，使采用 Java 开发的软件更加方便地执行和发布，避免了采用批处理文件进行启动带来的麻烦。通过此软件生成的 Exe 可执行文件可以自动从注册表、环境变量或者配置文件找到运行环境，并自动执行指定的启动类。

注意： jar2Exe 软件可以生成控制台程序、隐藏控制台的 Windows 窗口程序，以及后台启动运行的 Windows NT 服务程序 3 种类型的可执行文件。这时讲的是用此软件生成一个隐藏控制台的 Windows 窗口程序。

（1）安装完 jar2Exe 后，启动软件运行，然后输入 Jar 文件并指定 JRE 的版本，如图 12.60 所示。

（2）单击“下一步”按钮，选择要生成的程序类型，这里选择 Windows 窗口程序，如图 12.61 所示。

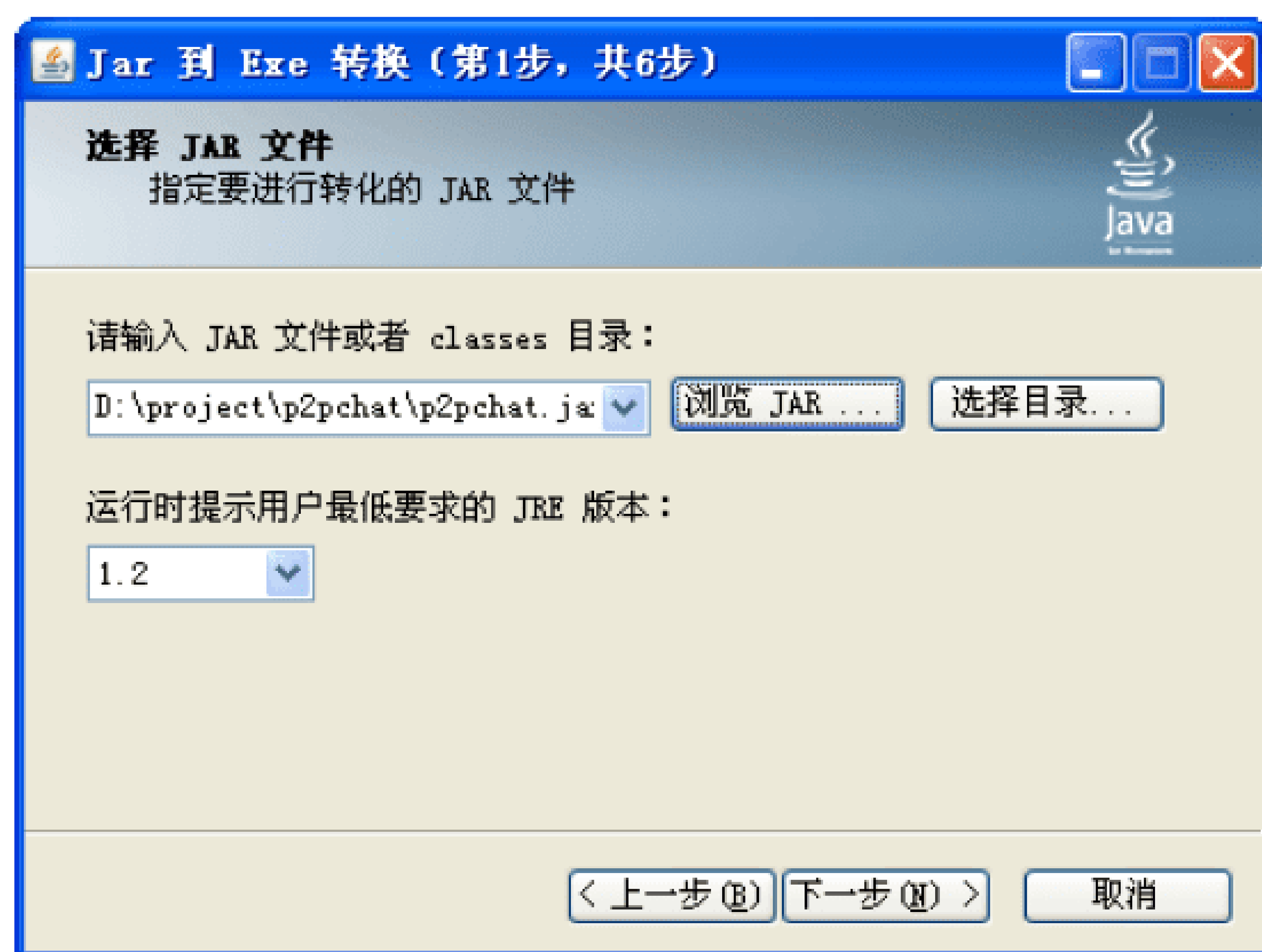


图 12.60　用 jar2Exe 软件导入 Jar 包的操作

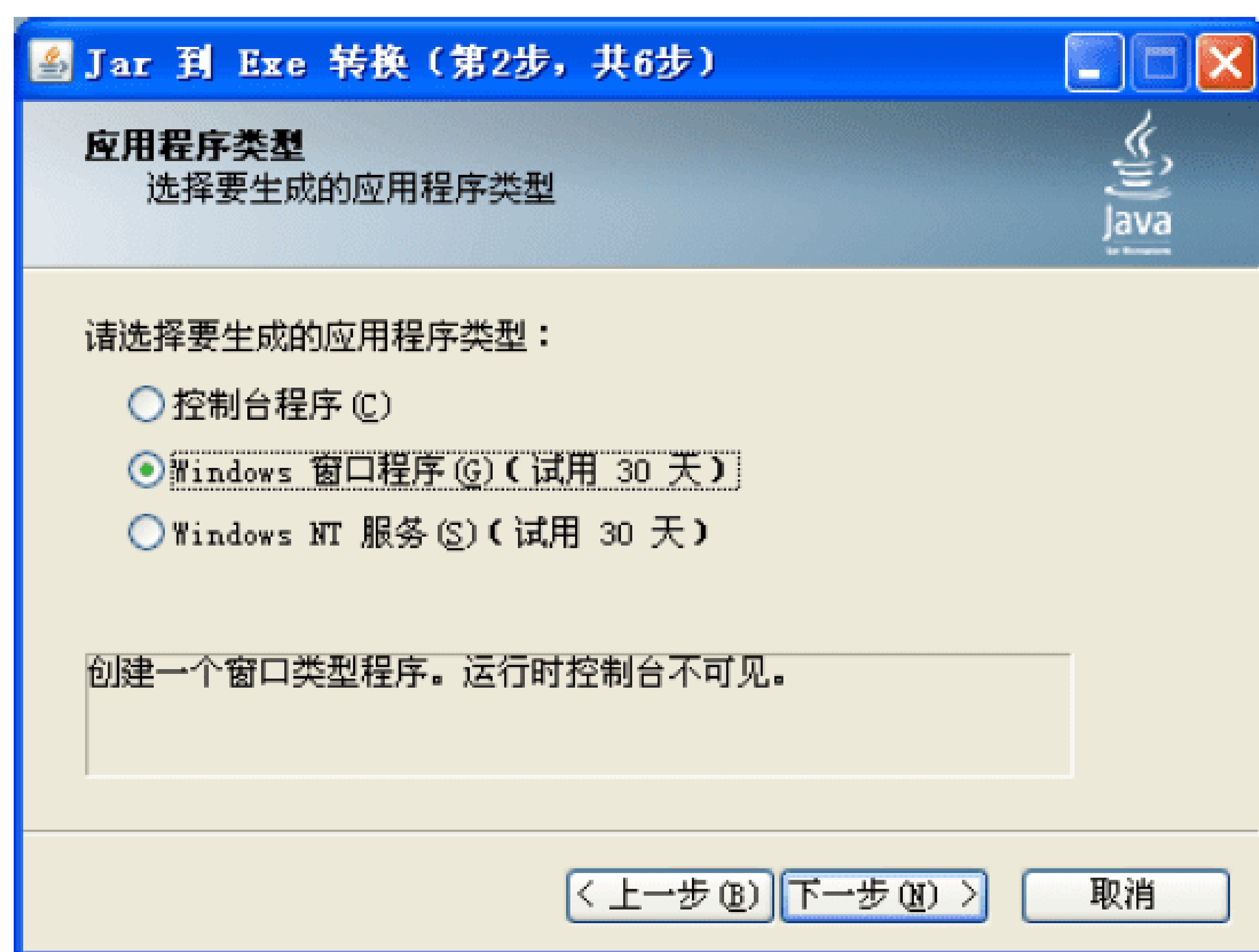


图 12.61　生成 Exe 文件类型的选择

注意： 本书中用到的是 jar2Exe 软件免费的非注册版本，读者在使用的过程中，如果仅作练习，也可以使用它的免费版，如有特殊需要，请购买正版。

（3）单击“下一步”按钮，在这一步中，选择系统启动的主类，也可以选择一张系统启动时的效果图片（这主要是为增强及美化启动效果而设置的），这里就省略了启动窗口图片，如图 12.62 所示。

（4）单击“下一步”按钮，在此步的设置中，主要是设置程序运行时的一些属性，是否显示任务栏图标、是否记录日志、源文件是否加密等，用户可根据需要自行选择，如图 12.63 所示。

（5）单击“下一步”后进入如图 12.64 所示的对话框。在这一步中，主要是添加工程中的依赖库。因为本系统不依赖任何第三方的 Jar 文件，所以不需进行设置，直接单击“下一步”，进入图 12.65 所示的对话框。

（6）在图 12.65 所示的对话框中，可设置运行程序的名字以及程序的版本、图标等信

息，单击“图标及版本（I）（试用 30 天）”按钮，弹出如图 12.66 所示的对话框。选择一个类似于 QQ 的小图标，设定版本为 1.0，直接单击 OK 按钮即可。然后再单击图 12.66 中的“下一步”，这时就会弹出如图 12.67 所示的提示消息框。

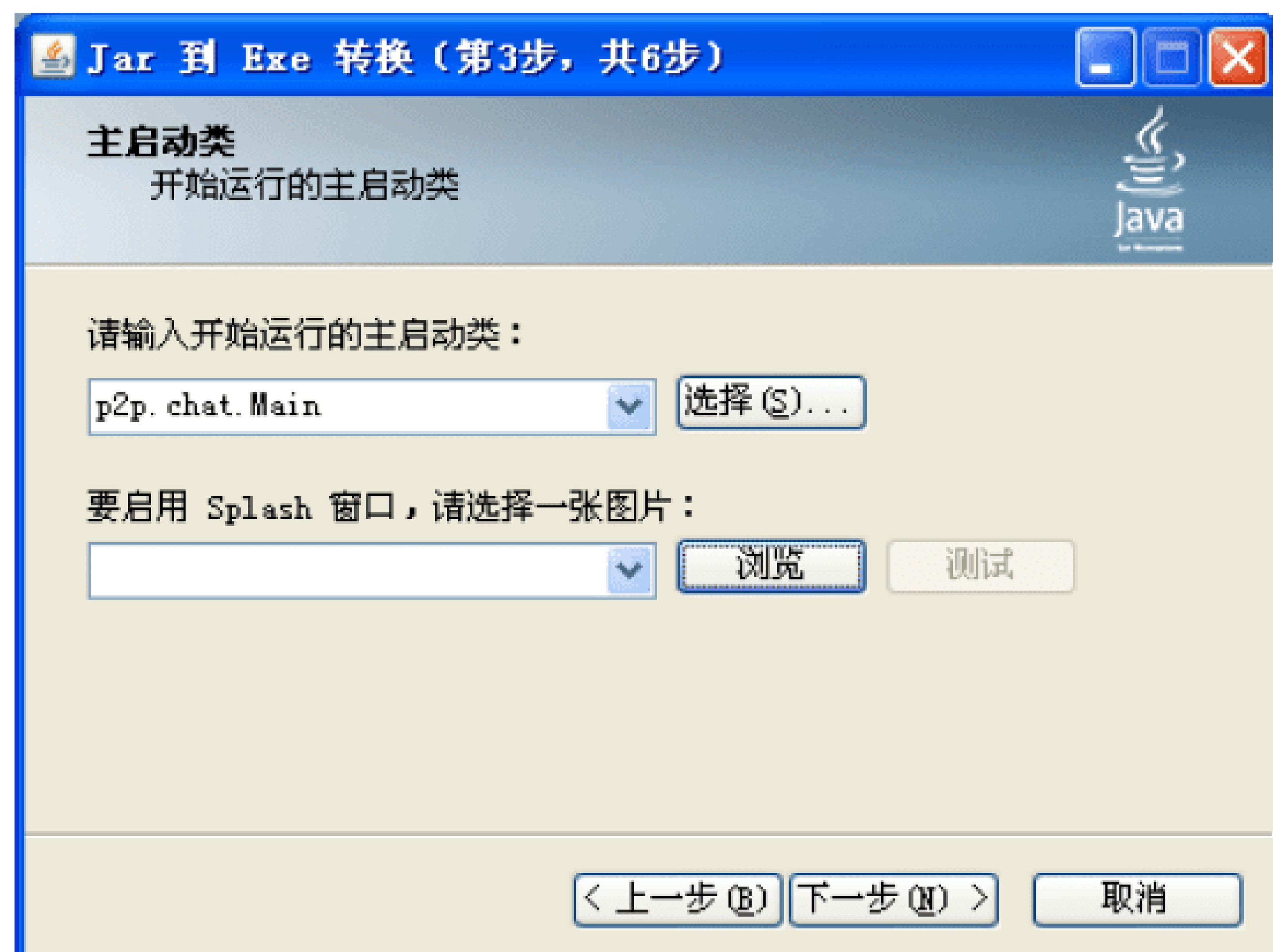


图 12.62　设置系统运行的主启动类

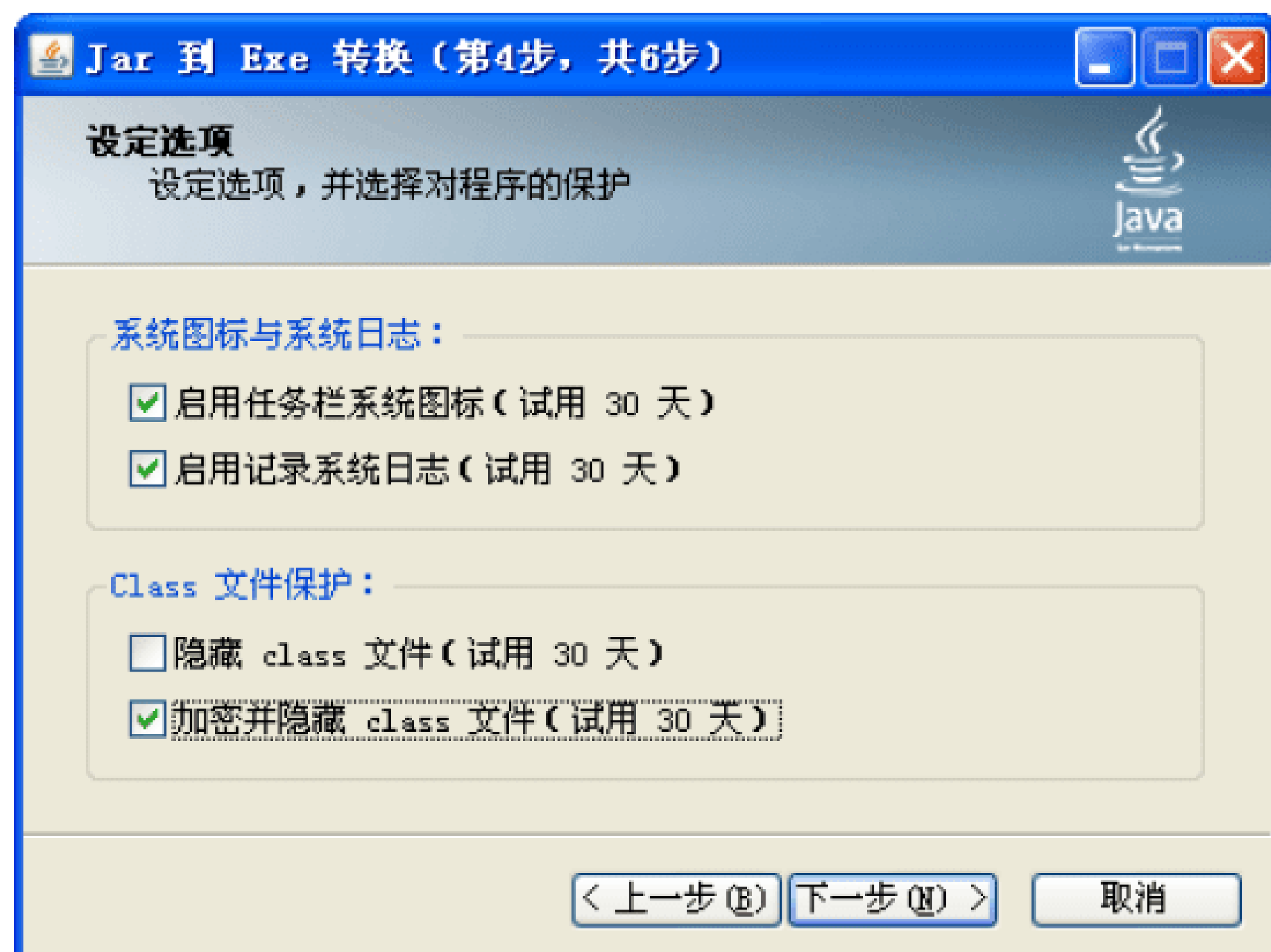


图 12.63　程序属性的设置界面

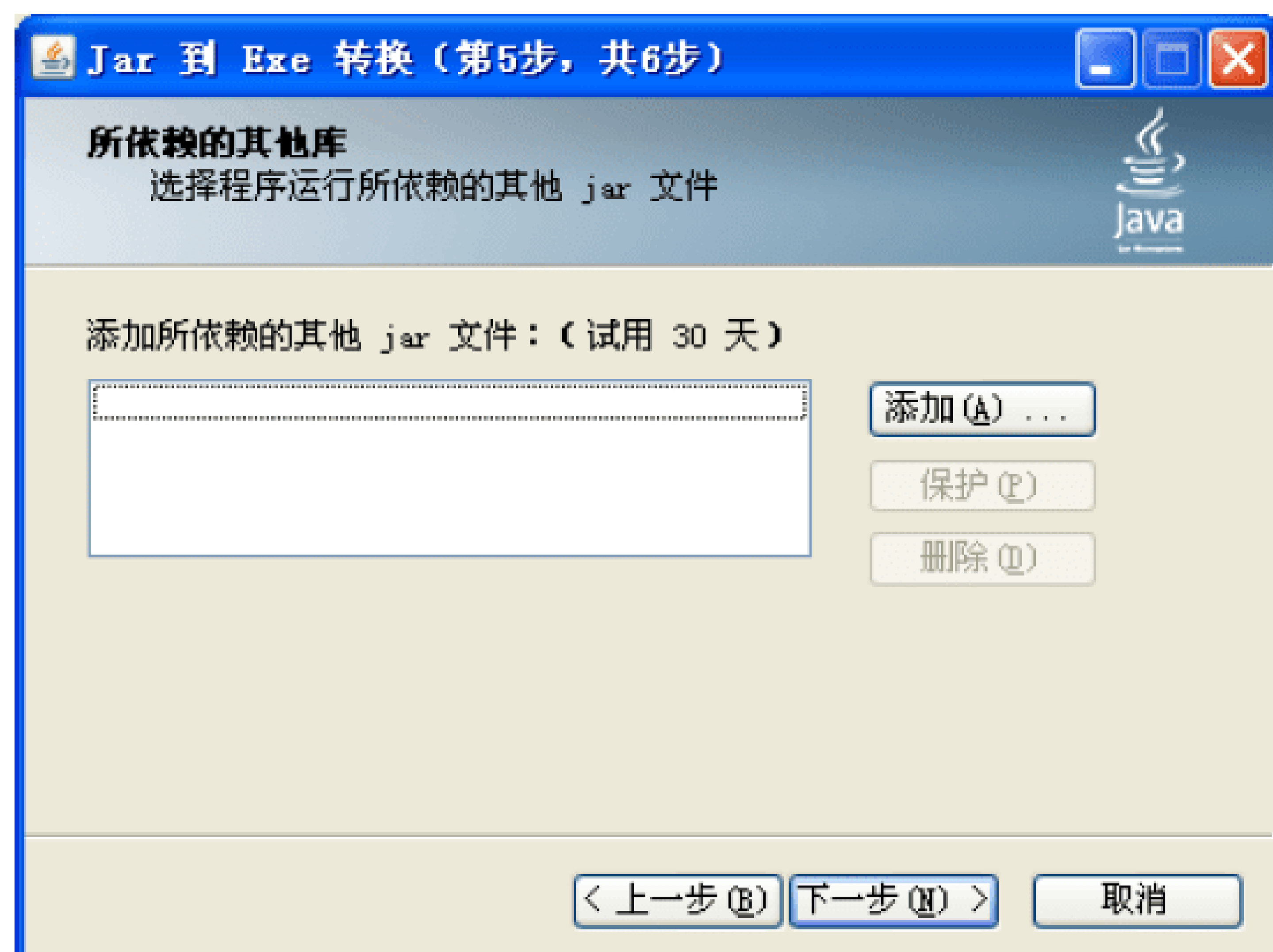


图 12.64　Fat jar 中引入依赖包的界面

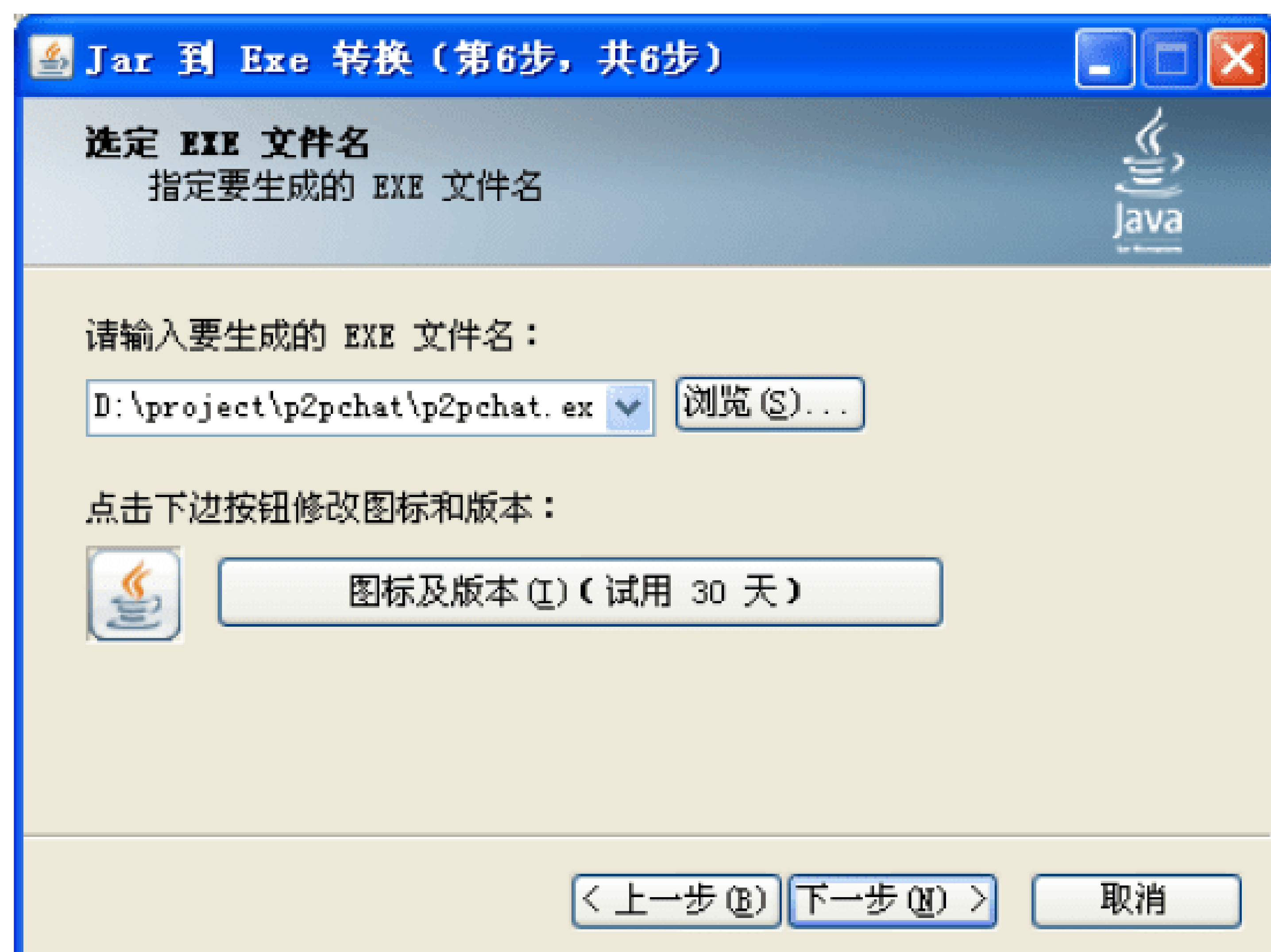


图 12.65　设置生成 exe 文件的位置

（7）在图 12.67 中，显示可执行文件生成成功，同时还可以进行参数配置，也可以直接打开文件夹，这里读者可自己去操作，单击完成，p2pchat 工程的 Exe 文件生成成功了。如图 12.68 所示，在 D:\project\p2pchat\目录下就生成了一个名为 p2pchat.exe 的可执行文件。

在图 12.68 所示的文件列表中，直接双击这个 p2pchat.exe 文件，就可以启动 P2P 的即时通信系统运行。

通过 Jar 或者 Exe 的方法打包发布 Java 工程的方法就讲完了。在实际开发中，打包 Java 工程的工具很多，生成 exe 文件的工具也很多，像 exe4j 这个软件，它可以把 Jre 的运行环境连同 Java 程序一同打包，这样在没有安装 Jre 的主机上，也可以直接双击运行程序。这对 Java 系统的发布而言是很方便的。对于一些大型的工程文件，还可以用 Inno Setup 软件制作打包安装程序，再进行发布，这样就更接近实际的开发应用了。

以上所讲的都是 Java 工程进行发布的最简单的方法，对于大型工程的发布管理，读者还需要在实践中不断地学习和深化。



图 12.66　设置运行程序的版本图标等信息

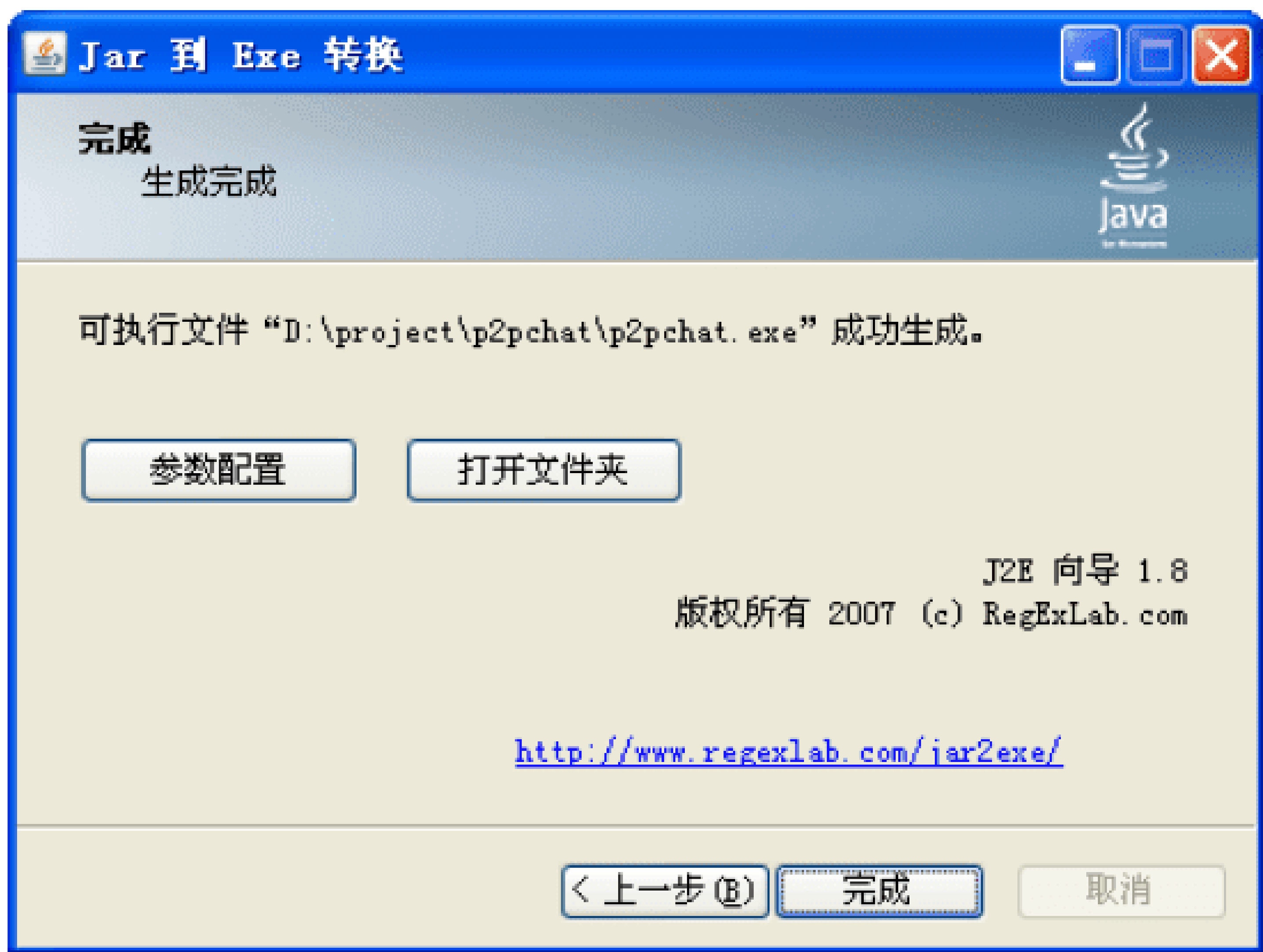


图 12.67　通过 Fat jar 成功生成 exe 文件

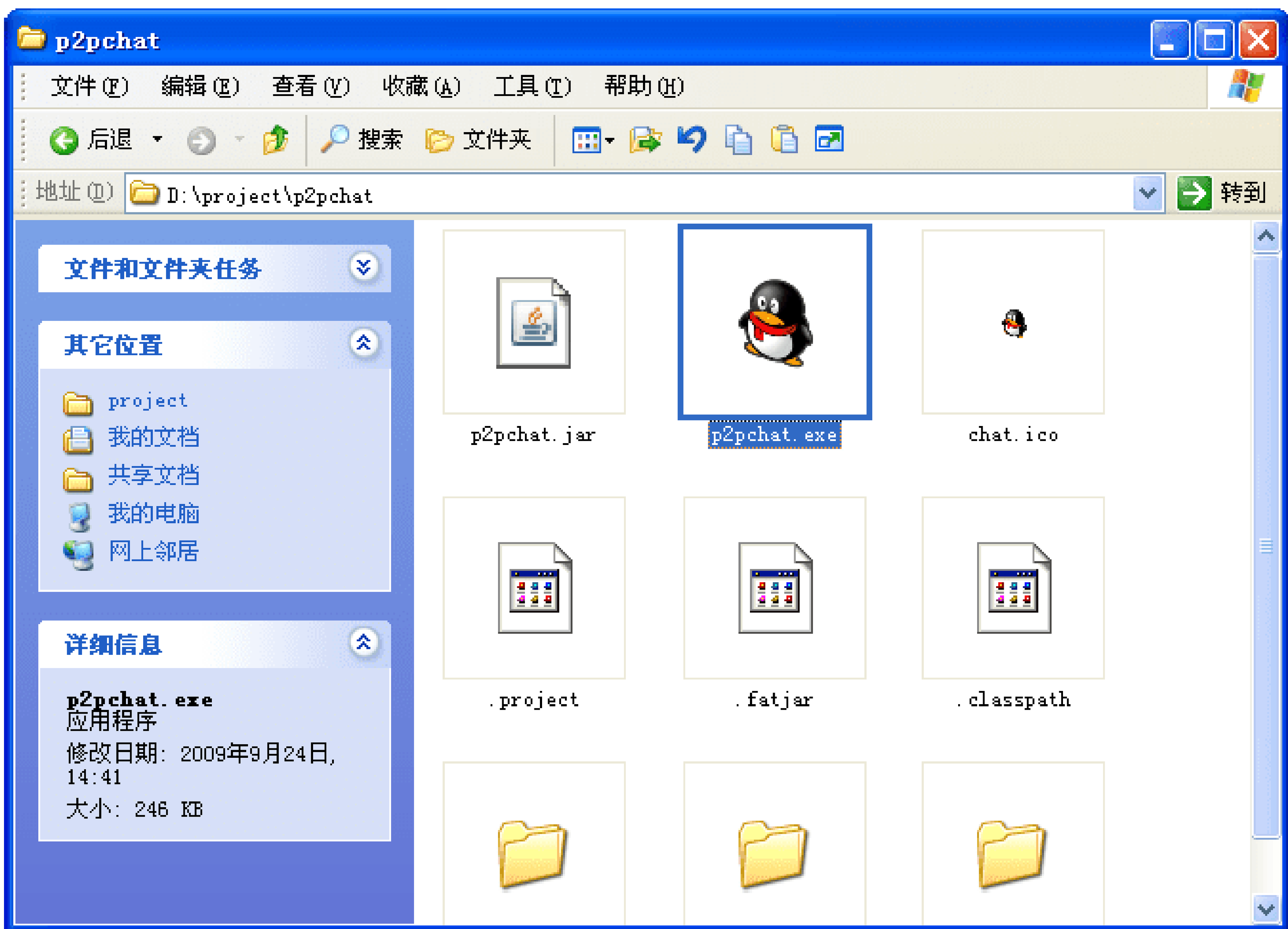


图 12.68　系统 exe 文件生成的效果图

12.8　系统开发的综合点评

本系统是一个基于 P2P 的即时通信系统，从最终的测试结果来看，基本上实现了这一系统的预期开发目标，完成了一些简单的 P2P 的底层实现机制和即时通信系统的基本功能，尤其是成功地完成了通过 IP 多播技术来实现网络中结点的发现和基本消息的交互，这在本章的学习中应该作为重点知识来掌握。

当然，本系统还有很多不完善的地方，如功能单一、不能完全体现 P2P 的特性、某些设计结构不合理等，这些都是需要进一步的改进，读者可以在本系统的基础上进行扩展开发，以进一步完善其功能和特性。

基于 P2P 的即时通信系统，作为一个学习的实例，其最终目的还是让读者进一步理解 P2P、熟悉 P2P、了解如何开发 P2P 的基本应用……这些才是最重要的。所以，在学习的过程中要与本书前面所讲的 P2P 的基本知识、即时通信的基本特点结合起来学习，这样，就更容易加深对所学知识的理解。

12.9　本 章 小 结

本章以实例的形式从需求分析—结构设计—类的设计—代码开发—系统测试—程序打包与发布这样一个完整的开发过程，一步步地带领读者开发一个系统的基于 P2P 的应用程序，重点在于让读者了解项目的基本开发过程，了解一些简单的 P2P 实现机制，了解 P2P 即时通信系统的工作原理和基本功能。学完本章还应该对 Java 语言的编程知识如 Java 网络编程、JavaGUI 编程、Java 加密编程以及 Java I/O 流知识等有更深的认识。

通过学习本章，读者不仅可以学习到 Java 项目开发中许多重要的知识，而且对简单项目开发的基本流程、模块划分和类的基本设计方法也会有所了解，其实。只要读者明白了了 P2P 即时通信系统的基本功能需求和实现的机制，在熟练的编程技术条件下，就完全可以独立地用任何语言开发出另一个 P2P 即时通信系统。

第 13 章　BT 系统分析及客户端开发方法

在本书的第 6 章中，对 BT 的工作原理及 BT 的协议规范进行了详细的讲解，而一个完整的 BT 系统就是在 BT 协议规范的基础上进行综合设计、系统分析，同时整合工作流程、实现程序算法而设计开发出来的。当前，基于 BT 的系统客户端有很多，几乎每一种语言都有相应的实现原型，如 Python、C++、Java 等，作为 P2P 技术的典型代表和经典应用，进行与 BT 有关的应用开发，不仅有现实的市场需求，对深入学习 P2P 技术也有重要意义。

注意：本文中出现的 BT 是 BitTorrent 的简写意思。

本章将从理论上分析一下 BT 系统模型，系统地了解一下来如何来构造一个 BT 系统模型，然后从实践的角度来讲解一下 BT 客户端的一般开发方法，以使读者进一步理解 BT 的一整套知识体系。

本章的主要知识点如下。

- BT 模型系统概述：了解 BT 系统的组成、执行流程及各实体元素之间的关系。
- BT 模型的运行过程：了解 BT 系统的一般执行过程，重点掌握 Peer 结点与 Tracker 服务器的交互过程及 Peer 结点之间的通信过程。
- BT 客户端的开发方法：了解 BT 客户端系统的功能和主要工作过程，掌握解析 torrent 文件的方法、掌握 Peer 结点与 Tracker 服务器的通信方法、掌握 Peer 结点之间进行交互的协议，并熟练运用编程来实现以上的过程。

13.1　BT 模型系统概述

BT 模型系统是一个理论上的 BT 系统，它可以完整地描述整个 BT 系统所需的基本元素、工作原理及过程等信息。

13.1.1　系统的组成

BT 是一种分发文件的协议。它通过 URL 来识别内容，并且可以无缝地和 Web 进行交互。它基于 HTTP 协议，其优势是，如果有多个下载者并发地下载同一个文件，那么，每个下载者也同时为其他下载者上传文件，这样，源文件可以支持大量的用户进行下载，而只带来适当的负载的增长。

注意： 在 BT 系统中，要分发某一个文件时，虽然系统有中大量需要分享此文件的用户，但并不会造成负载的增长，是因为大量的负载被均衡到整个系统中，所以提供源文件机器的负载只有少量增长。

在 BT 系统中，除了一般的结点之外，还有一个 Tracker 服务器，整个系统的结构模型如图 13.1 所示。

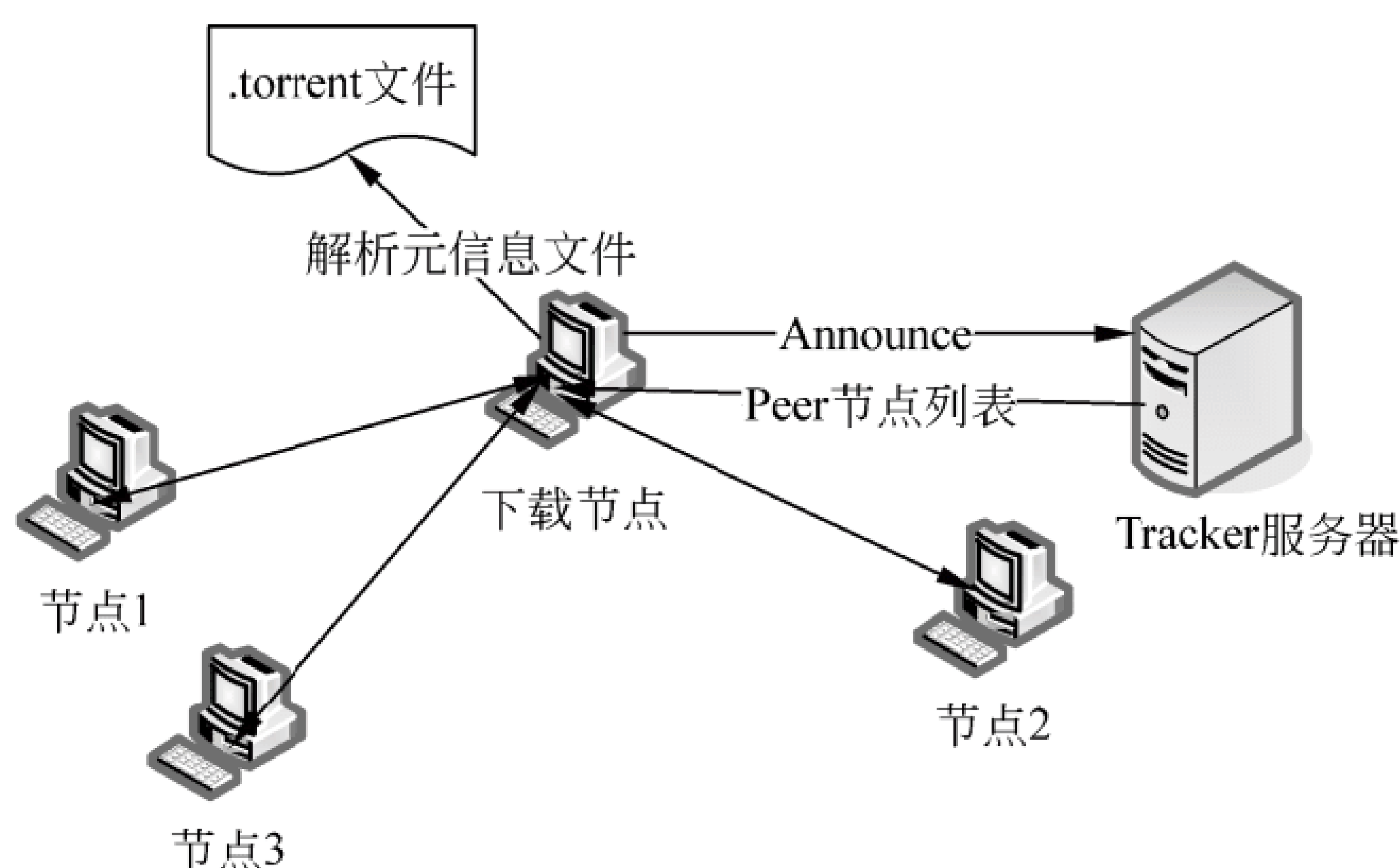


图 13.1　BT 系统模型结构图

图 13.1 中，Tracker 服务器记录了 BT 系统中所有的结点信息，可以协助新结点找到其他结点。用户如果要加入 BT 系统，需要先连上 Tracker，得到其他结点的信息，然后开始 P2P 通信。在此之前，用户必须先下载扩展名为.torrent 的文件，这个文件中记录了 Tracker 地址、要下载的文件摘要及一些其他的信息。

BT 下载中的所有实体及实体之间，为了完成文件下载功能而进行的交互中所遵循的技术规范，统称为 BT 模型。一个 BT 文件分布系统模型由下列实体组成：

- 一个普通的 Web 服务器，用于发布“元信息”文件。
- 一个静态的“元信息”文件，就是种子文件，即.torrent 文件。
- 一个跟踪（tracker）服务器。
- 终端用户的 Web 浏览器，用于下载种子。
- 一个文件提供者，被称为“种子”（seed），也叫源结点或种子结点。
- 终端下载者，BT 客户端，也叫运行 BT 客户端程序的下载终端，还可以称为 BT 网络中的结点（peers）

13.1.2　系统的执行流程

在 BT 模型系统中，要提供文件共享，必须要执行一系列的操作，这些操作有着一定的标准流程和顺序。在正常的情况下，完整地执行一个 BT 系统，一般分为两个大的过程，过程一：将文件发布出去；过程二：将文件下载下来。下面分别说一下这两个过程的详细执行步骤：

1．将文件发布出去

发布文件，指的是，在指定了 Tracker 服务器的基础上，将要发布的文件编码成“元

信息”文件，再将此“元文件”发布到网络中的过程，这一过程的详细执行步骤如下。

- 运行一个 tracker 服务器（或者，已经有一个 tracker 服务器在运行了也可以）。
- 运行一个 Web 服务器，例如 apache，或者已经有一个 Web 服务器在运行了。
- 在 Web 服务器上，将文件扩展名.torrent 和 MIME 类型 application/x-BT 关联起来（或者已经关联了）。
- 根据 tracker 服务器的 URL 和要共享的文件来创建一个“元信息”文件（.torrent）。
- 将“元信息”文件发布到 Web 服务器上
- 在某个 Web 页面上，添加一个到“元信息”文件的链接。
- 运行一个已经拥有完整文件的下载者（被称为“origin”，或者“seed”，种子）。

2．将文件下载下来

下载文件，就是 peer 结点利用 BT 客户端将发布在 Web 网络中用“元信息”表示的文件下载到本地的过程，这一过程主要在终端用户的 BT 客户端中完成，需要执行以下步骤：

- 安装 BT（或者已经安装）。
- 访问提供.torrent 文件的 Web 服务器。
- 单击到.torrent 文件的链接（编者注：这时候，BT 会弹出一个对话框）。
- 选择要把下载的文件保存到哪里？或者是一次断点续传。
- 等待下载的完成。
- 结束 BT 程序的运行（如果不主动结束，那么 BT 会一直为其他人提供文件上传）。

13.1.3　BT 系统中各实体之间的联系

在 BT 系统中，Peer 之间是通过 Tracker 服务器联系起来的，而 Peer 与 Tracker 之间联系的信息则是由.torrent 文件提供的，图 13.2 中简单的展示了它们之间的关系。

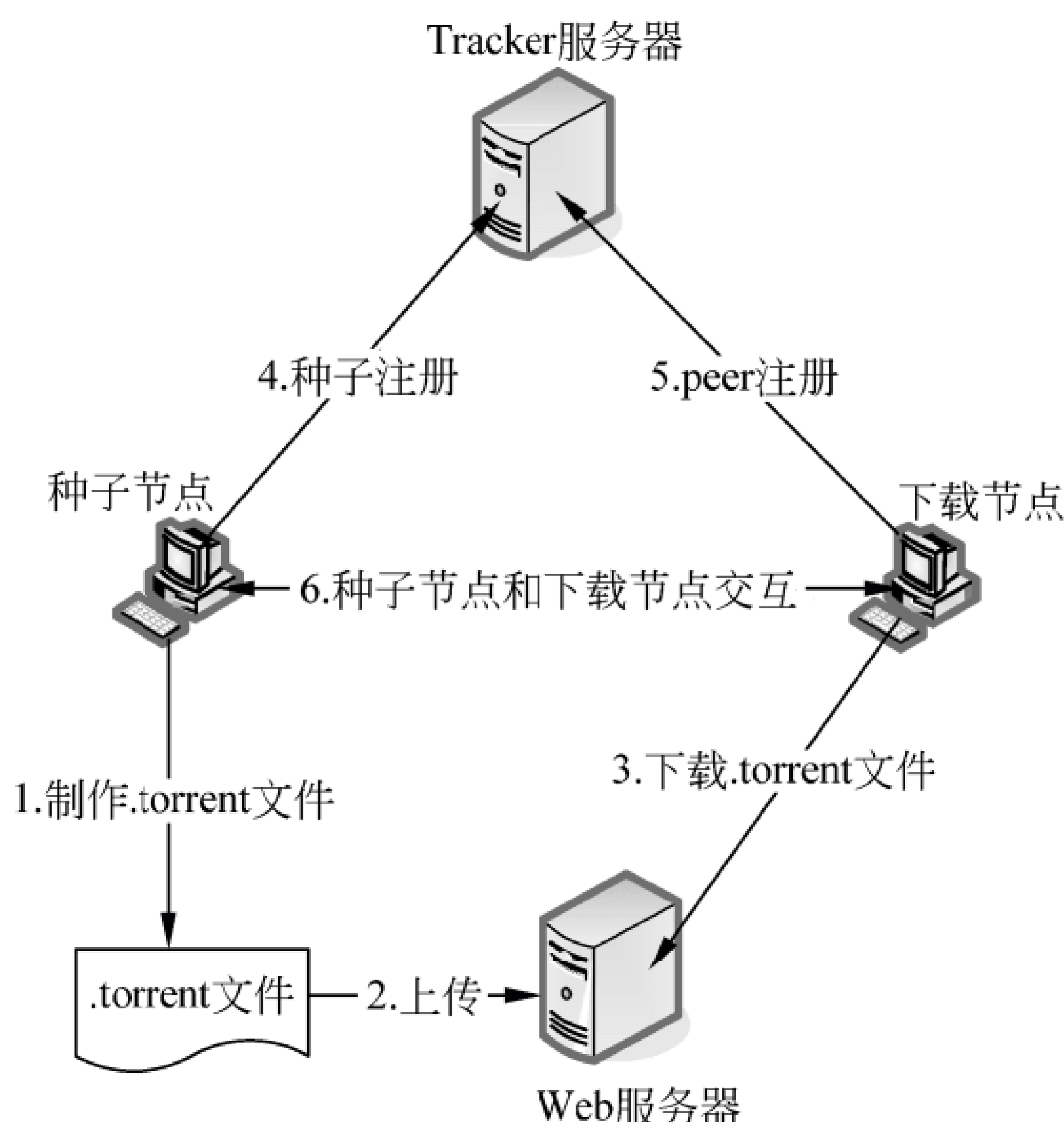


图 13.2　BT 系统中各实体之间的关系

在图 13.2 中，种子结点将制作好的描述发布文件信息的元信息文件（.torrent 文件）发布在 Web 服务器上。作为一个静态的文本文件，下载结点的用户可以自由下载此.torrent 文件，在下载结点所在的机器上安装的 BT 客户端程序可以解析此种子文件。

Tracker 从所有下载者（peer）处接收它们的注册信息，并返回给它们一个随机的 peers 的列表。这种交互是通过 HTTP 或 HTTPS 协议来完成的。

下载结点（作为下载者的 BT 客户端）周期性地向 tracker 登记，使得 tracker 能了解它们的进度；下载者之间通过直接连接进行数据的上传和下载。这种连接使用的是 BT 对等协议，它基于 TCP。

种子结点（Origin），首先得向 Tracker 服务器注册自己的存在，在文件的交互过程中，它只负责上传，从不下载，因为它已经拥有了完整的文件。要完成一个完整文件的下载，种子结点是必须的。

13.2 BT 模型的运行过程

一个完整的 BT 系统的工作过程，可以分为 4 个大的步骤，这 4 步分别是：文件信息发布、追踪服务器的部署、与 Tracker 交互取得 Peer 信息、各 Peer 结点之间的交互式下载。这 3 个过程是任何一个 BT 系统都必须经历的。下面分别讲这 3 个步骤的具体运行过程。

13.2.1 文件信息的发布

文件信息发布，在上文中已经说过，就是将你要发布的文件编码成“元信息”文件，再将此文件发布到网络中，这就完成了文件信息的发布。

在具体的发布过程中，需要注意几点。

- 元信息文件：需要将原始结点将文件的基本信息以及追踪服务器的 URI 创建一个元信息文件，所以，在创建这个元信息文件（.torrent）的时候，需要部署一个 Tracker 服务器，或已知一个可用的 Tracker 服务器，这是元信息文件里必须要有的内容。
- 发布：元信息文件需要发布到 Web 网络中，并提供一个有效的链接，这个过程需要一个 Web 服务器，或已知一个可用的 Web 服务器。
- 种子：种子就是原始的结点，它必须存在，且是运行着 BT 客户端程序的活跃结点，以便其他结点能够与其建立连接。

注意： *原始结点和其他结点的角色是相同的，所不同的是只上传数据，并不下载，所以被称为“种子”。*

以上就是关于文件信息发布的过程，在实际应用中，文件信息的发布并不是我们关注的重点，因为，当前有无数的 BT 种子发布网站，有各种现成的.torrent 文件，所以，读者只需知道这个过程就可以了。

13.2.2 追踪服务器的部署

BT 下载过程中需要一个追踪服务器。追踪服务器负责帮助结点获取其他结点的信息，

并协调不同结点之间的信息。追踪服务器和结点之间使用 HTTP 协议进行交互，结点向追踪服务器注册下载的文件、IP 地址、端口以及相关信息，追踪服务器告诉结点其他结点的注册信息，结点利用这些信息相互之间建立连接。

BT 下载模型部署的第一步就是建立并运行一个追踪服务器。追踪服务器必须有一个形如 http://域名（IP 地址）:端口号/announce 的 HTTP 链接。这个链接也规定与服务器的通信协议。

在与服务器进行通信的过程中，追踪服务器接受结点的 HTTP GET 请求，请求包括来自结点的定期“注册”，这些“注册”消息使得追踪服务器拥有关于下载结点的所有统计信息。追踪服务器的响应的信息，包括一个结点列表，这个列表中的信息使得结点之间能够相互通信并彼此参与下载。

13.2.3　与 Tracker 交互

结点与 Tracker 交互的过程是这样的，结点向 Tracker 服务器发送一个 HTTP 的 GET 请求，此请求的目的是向服务器索要当前可用的结点列表。服务器根据请求的方式，返回相应的应答消息，在应答消息里有其他的 Peer 结点的信息，根据这些信息 Peer 之间就可以建立连接，相互共享资源了。

整个交互过程，要完成两个工作，一个是请求，另一个是应答。

1．请求过程

请求过程由是 Peer 发送的一个 HTTP 的 GET 请求来完成的，Peer 发给追踪服务器的 GET 请求中的参数如下。

- ❑ info - hash：元信息文件中 info 关键字/值的 20 个字符的 SHA1 散列值。
- ❑ peer- id：标识结点唯一的 20 个字节的字符串。
- ❑ port：结点监听的端口号。
- ❑ up loaded：上传的字节数。
- ❑ downloaded：下载的字节数。
- ❑ left：还需要下载的字节数。
- ❑ event：值必须是 started、comp leted、stoppe 其中之一，或者为空，等同于未使用。
- ❑ ip：指明客户端机器的真实 IP 地址。
- ❑ numwant：可选参数，指明结点希望从追踪服务器得到的其他结点的个数。默认值通常是 50。

2．应答过程

应答就是服务器根据 Peer 请求的方式，返回给 Peer 一个响应消息，服务器给结点的响应消息的结构用伪代码表示如下：

```
union answer- msg {
//请求失败原因的错误消息，若该关键字有效，则其他关键字就不需要
char3 failure- reason;
struct success {          //请求成功时返回的消息结构
int interval;             //结点发送请求间隔秒数
```

```
int comp lete;            //“种子”结点的个数
int imcomp lete;          //非“种子”结点的个数
char3 3 peers;            //描述结点信息的字符串数组
int port;                 //结点使用的端口号
}
```

注意：下文会有相应的程序实现。

13.2.4　结点之间的交互式下载

结点从普通 Web 服务器获得元信息文件，通过该文件获得追踪服务器的 URI，根据该 URI 向追踪服务器注册，并从追踪服务器得到其他结点（包括“种子”结点）的注册信息。结点根据获得的信息与其他结点之间建立连接，开始下载和上传数据。

BT 下载方式中，将下载文件划分为若干个片断（pieces），结点下载文件时，选择自己需要的某些片断进行下载，同时将自己拥有的片断提供给其他感兴趣的结点。当某个结点完成了所有文件片断的下载后，停止下载，但是上传仍然在继续。由于拥有一份完整的文件，所以这个结点也成为一个“种子”，一直到退出 BT 下载。

BT 下载模型中，完整的文件可以是由“种子”提供的，也可以是多个结点所拥有的片断的并集。

Peer 结点之间的交互主要依据 BT 协议规范中的结点线路协议（PeerWire Protocol）来完成，此协议是基于 TCP 协议的应用层协议，是结点之间进行数据交换所使用的协议。此协议主要定义了两个信息，一个是连接状态信息，另一个是握手信息。

1．连接状态信息

每个结点必须知道和其他结点之间每个连接的状态信息，包括下列两种状态：

- choked：远程结点是否“堵塞（ choked） ”与该结点的连接。 一个结点堵塞了该结点，该结点的任何请求都不会得到应答，直到被“疏通（ unchoked） ”。
- interested：远程结点“关注（ interested） ”该结点。 当连接被疏通时，远程结点即将开始请求获得文件。

注意：“阻塞”和“关注”都是双向的，不仅仅是某个结点针对其他结点的，其他结点也针对该结点。

2．握手（Handshake）消息

```
//“握手”必须是结点发送的第 1 个消息，此消息的伪代码定义如下：
struct Handshake {             //握手消息
int p strlen;                  //字符串 p str 的长度
char3 p str;                   //协议标识
char reserved[ 8 ];            //保留
char info- hash[ 20 ];         //结点的 SHA1 散列值
char peer- id[ 20 ];           //结点标识
}
```

13.3　BT 客户端的简介

BT 客户端是 BT 系统的重要组成部分，是进行 BT 文件下载的直接载体。当前有很多不同语言开发实现的 BT 客户端，其中有很多系统都是开源的，第一个 BT 客户端原型系统就是由 Python 语言开发而成的，它是一个开源系统，在网络上可以得到它的实现源代码。

上文中讲解了整个 BT 系统的模型以及 BT 客户端的一些功能和运行机制，那么要实现这个客户端系统也并非难事，本节就演示一下 BT 客户端系统的开发流程和一些重要算法的实现。

13.3.1　BT 客户端主要功能与协议

BT 客户端就是用于在 BT 网络中进行文件交互下载的一款软件工具，它是依据 BT 协议规范进行开发的，BT 客户端可以用任何一种编程语言来实现，也可以开发成任何风格，但它们的功能和使用协议都是相同的。

1．BT客户端的主要功能有以下几点

- 解析.torrent 文件，获取要下载的文件的详细信息，并在磁盘上创建存储空间。
- 与 tracker 服务器建立连接，并交互消息。
- 根据从 tracker 得到的信息，跟其他 peers 建立连接，并下载需要的文件片断。
- 监听某端口，等待其他 peers 的连接，并提供文件片断的上传。

2．BT客户端所用到的相关协议

对客户端来说，它需要处理两种协议：

- 与 tracker 服务器交互的 track HTTP 协议。
- 其他 peers 交互的 BT 对等协议。

BT 客户端的主要功能是所有的客户端系统都必须要实现的，它所涉及的协议也是在系统开发过程中必须遵循的。

13.3.2　BT 客户端的核心工作过程

作为一个普通的 BT 客户端，它要完成一次 BT 文件的下载，它应该有如下几个核心的过程。

1．解析.torrent文件

通过 BT 客户端来下载一个资源的话，首先要有种子文件，也就是.torrent 文件，关于.torrent 文件的编码方式、内容要求等在本书的第 6 章已有详细的说明，这里就不再赘述。当一个客户端拥有这个.torrent 文件后，第一件事要做的就是解析这个.torrent 文件，将此文

件描述的所有信息全部解析出来。

2. 与Tracker服务器通信

解析完.torrent 文件后，就能得到 tracker 的列表、info_hash、文件长度等所有的信息。再根据与 tracker 服务器交互的协议规定，通过 HTTP 的方式连接到 Tracker 服务器，从 tracker 服务器上得到 Peer 的 IP 地址和端口信息。

3. 与Peer之间进行文件的交互

得到 Peer 的 IP 地址、端口等信息后，就可以直接与 Peer 进行交互，再根据 Peer 间交互协议的规定，Peer 之间就可以进行文件的上传和下载了。

以上所描述的就是 BT 客户端的基本工作流程，分别通过解析种子文件、与 Tracker 交互、与 Peer 交互等，最终实现点到点之间数据的传输。这 3 个过程整合起来就构成了 BT 系统的整个工作流程，13.4 节就讲一下这 3 个过程的具体实现方法。

下文所要讲到的 BT 客户端的开发只是对其工作的核心过程进行描述并用代码实现，只提供一个开发的思路和方案，不再讲解每一个实现细节。因为就 BT 客户端的完整系统而言，它是一个非常大的工程，非此书所能涵盖，有些东西也都超出了 P2P 技术的范围。

注意： 对 BT 客户端系统感兴趣的读者，建议研究一下 Java 开发的开源的 Vuze 系统，它是 BT 客户端系统里开源软件的经典代表，一直活跃在开源社区的第一位。

13.4 解析.torrent 种子文件的实现方法

BT 协议中大部分信息传递是用的 B 编码规则进行编码过的数据，.torrent 文件也是一种 B 编码的文件，它存储了 Tracker、文件摘要等信息，解析.torrent 种子是一个 BT 客户端所必备的功能。

从实现方法上看，解析种子文件的过程就是将以 B 编码形式存储的.torrent 文件的内容全部读取出来的过程，在这个过程中需要根据 B 编码规则来编写相应的解析算法。

13.4.1 .torrent 文件的编码规则及说明

B 编码的数据格式、编码规则等已经讲过，这里再简单地复习一下。

1. B编码的格式与要求

B 编码支持 4 种格式，分别为字符串、数字、列表、字典。

对于字符串，首先是一个字符串的长度，然后是冒号，后面跟着实际的字符串，例如 4:spam，就是 spam。

对于整数，以 i 开始，然后是 10 进制的整数值，最后以 e 结尾。例如，i3e 表示 3，I-3e 表示-3。其中 0 用 i0e 表示。

对于列表的编码，以 l 开始，接下来是列表值的编码（也采用 bencoded 编码），最后以 e 结束。例如 l4:spam4:eggse 表示['spam'，　'eggs']。

对于字典编码，以 d 开始，接下来是可选的 keys 和它对应的值，最后以 e 结束。例如 d3:cow3:moo4:spam4:eggse 表示{'cow':'moo'，'spam':'eggs'}，而 d4:spaml1:al:bee 表示{'spam':['a'，'b']}。键值必须是字符串，而且已经排序（并非是按照字母顺序排序，而是根据原始的字符串进行排序）。

2．.torrent文件的结构

.torrent 文件是一个字典结构，包括以下几个关键字。

- info 字典结构，里面描述了文件的一些信息，　包括两种情况，一是单个文件，再一个就是包括目录的多个文件。
- announce string　结构，tracker　的发布地址。
- announce-list　列表，　多个 tracker　地址。
- creation date　数字（可选）发布时间。
- comment string　注释。
- created by string，发布者。

3．info信息结构

Info 本身也是一个字典结构，主要用来描述文件的摘要信息，info 字典根据所描述的多文件还是单文件的不同其结构也不同，具体的结构描述如下：

（1）下面部分在单个文件与多个文件中都存在

- piece length：整数，每块的大小。
- pieces：整数，块数。
- private：整数，如果设置为 1，客户端必要从 metafile　文件中的 tracker 得到种子，如果没有设置或设置为 0，则可从任何途径获得种子。

（2）描述单个文件

- name：字符串，文件名。
- length：整数，文件大小。
- md5sum：可选，文件的 md5 值，32 个字符。

（3）描述多个文件

- name：文件夹的名字。
- files：一个字典组成的列表，每一项有以下关键字。
- length：整数，文件大小。
- md5sum：字符串，可选。
- path：路径，文件夹和文件名的 bencode 编码的列表。

总结以上所说的，整个.torrent 文件其实就是一个大字典，字典内部嵌套着各种不同的数据类型，包括字符串、数字、列表、字典等，.torrent 文件结构可用图 13.3 表示。

从图 13.3 所示的文件结构中，可以清楚地看到整个文件的类型、层次以及包含关系，还可以看出.torrent 文件所描述的文件信息。根据这种结构和数据之间的嵌套关系，再加上 B 编码的规范定义，就可以编写出相应的解析算法。

```
|Root(dict)
|--|announce(str)
|--|announce-list(list)
|--|--|0(list)
|--|--|--|0(str)
|--|--|--|1(str)
|--|created by(str)
|--|creation date(int)
|--|info(dict)
|--|--|length(int)
|--|--|name(str)
|--|--|name.utf-8(str)
|--|--|piece length(int)
|--|--|pieces(str)
```

图 13.3　.torrent 文件的整体结构图

13.4.2　解析.torrent 文件的实现代码

要解析一个用 B 编码规则定义的.torrent 文件，首先，定义一个 Torrent 类，这个类所描述的就是整个.torrent 文件的结构，在这个类的源代码中附有详细的注释说明。

Torrent 类位于 btclient_test 工程中的 p2p.btclient.torrent 包下，以下是 Torrent 类中主要方法的说明，详细完整的实现源代码请读者参考随书光盘，Torrent 类的文件位置如下：

【示例源代码：ch13\ch13_code\btclient_test\src\p2p\btclient\torrent\Torrent.java】

```
/**
  *Torrent 类，根据 B 编码规则的定义，来描述.torrent 文件的结果，以建立一个 torrent
  *文件结构模型，方便后续的操作。具体实现如下：
  */
package p2p.btclient.torrent;
import java.io.*;
import java.util.*;
import p2p.btclient.TorrentException;
import p2p.btclient.tracker.BencodeUtils;
public class Torrent {
    //得到本系统的文件路径的分隔符
    private static String Separator = System.getProperty("file.separ-
    ator");
    //定义 torrent 文件中几个基本的属性
    private String announce;             //字符串形式的 annuounce
    private List<String> announceList; //列表形式的 annuounce_list
    private Date creationDate;
                            //创建日期（可选），torrent 文件是以长整形数据存储的
    private String comment;              //字符串形式的 comment
    private String createdBy;            //字符串形式表示此 torrent 文件的作者
    private Map info;                    //定义 Info 的结构，字典型的，用 Map 表示
    private boolean noAnnounceList = false;    //用于判断是否有 AnnounceList
    //torrent 文件中，info 也表示的是个字典，文件内容信息也存储在这个结构里，所以用一
    个 InfoContent 类来专门表示 info 的内容
    private InfoContent infoContent;
    //整个 torrent 文件就是一个字典结构，所以定义一个描述整个结构的 Map 型 fullInfo，
```

```
    用于存放 torrent 文件的全部内容
    private Map fullInfo;
    /* 在 BT 协议中规定，info_hash，是 URL 编码的 20 字节 SHA1 散列，这个散列是元信息
    文件中 info 键所对应的值的 SHA1 散列，这里直接将其定义为一个属性，在后面进行访问
    tracker 的操作时需要用到这个值。
     */
    private byte InfoHash[];
    //如果列表是 announce，则将其存放在一个字符串数组里
    private String[] listAnnounce;
    //构造方法，接收文件作为参数
    public Torrent(File file) throws IOException {
        this(BencodeUtils.readCompleteFile(file));
    }
    //构造方法，接收字节数组作为参数
    public Torrent(byte abyte0[]) {
        //调用 BCode 类的 BDecode 解码方法，得到一个 Map 型的整个 torrent 文件的内容
        fullInfo = (Map) BCode.BDecode(abyte0);
        //根据 key='info'，从 fullInfo 这个 Map 中得到 info 的值
        info = (Map) fullInfo.get("info");
        //infohash 是对 info 字典内容进行 B 编码后，再由 SHA1 函数进行散列得到的，Bcode
        的 BEncode 方法是进行 B 编码，BencodeUtils 的 shaHash()方法，是将编码内容进
        行 SHA1 散列
        InfoHash = BencodeUtils.shaHash(BCode.BEncode(info));
        //根据 key='announce'，从 fullInfo 这个 Map 中得到 announce 的值
        announce = new String((byte[]) fullInfo.get("announce"));
        /*
         * 以下的标签在 BT 协议规范中，都是可选的，因而需要执行一个判断。
         */
        //如果有，则取出 announce-list 的值
        if (fullInfo.containsKey("announce-list")) {
            announceList = (List) fullInfo.get("announce-list");
        }else {
            this.setNoAnnounceList(true);
        }
        //如果有，则取出 comment 的值
        if (fullInfo.containsKey("comment")) {
            comment = new String((byte[]) fullInfo.get("comment"));
        }
        //如果有，则取出 creation date 的值
        if (fullInfo.containsKey("creation date")) {
            creationDate = new Date(((Number) fullInfo.get("creation
            date")).longValue() * 1000L);
        }
        //如果有，则取出 created by 的值
        if (fullInfo.containsKey("created by")) {
            createdBy = new String((byte[]) fullInfo.get("created by"));
        }
    }
```

根据对.torrent 文件规则的分析，在.torrent 文件中，info 标签所表示的内容也是以一个字典结构存储的，它存储了文件的主要信息，所以还需要对 info 标签进行进一步的处理，以下就是处理 info 标签信息的代码实现。

```
/**
 * Method name: getTorrentInfo
 * TODO : 将 info 所表示的字典内容再进行进一步的处理，info 字典里表示的都是文件信息，
```

```
这里就是处理这些信息的过程
 * return_type:InfoContent
 */
public InfoContent getTorrentInfo() {
    info = this.getInfo();
    long fileLength;
    long singleFileLength;
    String md5Sum;
    //首先要进行判断，确保所要处理的内容不为空
    if (infoContent == null) {
        infoContent = new InfoContent();
        if (info == null) {
            throw new TorrentException("在此 torrent 文件中，没有发现'info'字
            典！");
        }
        //根据 B 编码中定义的 info 的结构，分别取出相应的值
        String name = new String((byte[]) info.get("name"));
        Integer pieceLength = ((Number) info.get("piece length")).int-
        Value();
        String pieces = new String((byte[]) info.get("pieces"));

        //这 3 个字段在 info 字典中都是必须的，所以需要进行一次判断，如果没有这些键值
        证明 torrent 文件存在问题。出现问题时抛出自定义的 TorrentException 异常
        if (name == null) {
            throw new TorrentException("在 info 字典中没有找到键值 name");
        }
        if (pieceLength == null) {
            throw new TorrentException("在 info 字典中没有找到键值 piece-
            Length");
        }
        if (pieces == null) {
            throw new TorrentException("在 info 字典中没有找到键值 pieces");
        }
        //将取得的值 set 到 infoContent 里，在读取文件信息的时候会用到
        infoContent.setName(name);
        infoContent.setPieceLength(pieceLength.intValue());
        infoContent.setPieces(pieces);
//下面这段代码就是读取文件信息的核心代码，首先根据 filelist 来判断是多文件还是单文件
        List filesList = null;
        if (info.containsKey("files")) {
            filesList = (List) info.get("files");
        }
//如果 torrent 文件信息为多文件，则单独用一个 MultiFile 类来表示，MultiFile 数组用来
存储多文件里所有的文件
        MultiFile files[];
        if (filesList == null) {
// 处理单文件的内容，根据 BT 协议规范，单文件有 3 个键，分别为 name、length 和 md5sum
            if (info.containsKey("name")) {
                String filename = new String((byte[]) info.get("name"));
                infoContent.setName(filename);
            }//在 if 后面，也可以用一个 else 语句打印一条提示信息，或进行其他操作均可
            if (info.containsKey("length")) {
                fileLength = ((Number) info.get("length")).longValue();
                infoContent.setFileLength(fileLength);
            }
            if (info.containsKey("md5sum")) {
                md5Sum = new String((byte[]) info.get("md5sum"));
                infoContent.setMd5Sum(md5Sum);
            }
```

```
            //标识一下，info 字典里存储的是单文件
            infoContent.setSingleFile(true);
        } else {
//处理多文件的过程，根据 BT 协议规范，多文件中也有 name、length、md5sum、path 等，以
下的处理方法，就是将这些信息解析出来
            //初始化一个 MultiFile 数组
            files = new MultiFile[filesList.size()];
            //通过 filesList 的 toArray()方法，取出所有的 Map
            Map fileListmap[] = (Map[]) filesList.toArray(new Map
            [filesList.size()]);
            //对所有 Map 结构进行遍历
            for (int i = 0; i < fileListmap.length; i++) {
                Map listMap = fileListmap[i];
                MultiFile torrentFile = new MultiFile();
                files[i] = torrentFile;
                //对 listMap 中包含的 Key 值进行判断，是否有值为 length 的 key
                if (listMap.containsKey("length")) {
                    //如果有，则取出 length 所对应的值，此值就是单个文件的大小
                    singleFileLength = ((Number) listMap.get("length")).
                    longValue();
                    //通过 torrentFile 对象的 setXX 方法，设置文件长度
                    torrentFile.setSingleFileLength(singleFileLength);
                }
                //对 listMap 中包含的 Key 值进行判断，是否有值为 md5sum 的 key
                if (listMap.containsKey("md5sum")) {
                    //如果有，则取出 md5sum 所对应的字符串
                    String md5sum = new String((byte[]) listMap.get("md5-
                    sum"));
                    //将得到的字符串设定为 md5sum 值
                    torrentFile.setMd5sum(md5sum);
                }
                //对 listMap 中包含的 Key 值进行判断，是否有值为 path 的 key
                if (listMap.containsKey("path")) {
                    //因为是多文件的，有多个文件路径，所以用 List 存储
                    List pathList = (List) listMap.get("path");
                    //初始化一个 path 的 StringBuffer 对象
                    StringBuffer path = new StringBuffer();
                    for (int j = 0; j < pathList.size(); j++) {
                        //在 path 中添加文件路径信息
                        path.append(new String((byte[]) pathList.get
                        (j)));
                        //在路径中加入目录分隔符
                        path.append("/");
                    }
                    //在 path 中，最后一个"/"分隔符是不需要的，所以去掉
                    path.deleteCharAt(path.length() - 1);
                    //将得到的 path 信息，设定到 torrentFile 对象中
                    torrentFile.setPath(path.toString());
                }
            }
            //将整个 files 设定到 infoContent 中
            infoContent.setMultiFile(files);
        }
    }
    //返回 infoContent
    return infoContent;
}
```

以上的代码块主要用来取得.torrent 文件中关于 Announce 的信息，通过 getListAnnounce()方法，取出所有的存储着的 Announce 信息。

因为 announce-list 的内容是一个 B 编码的列表，要取出字符串类型的值，还需依据 B 编码规则来实现，具体的实现代码如下：

```
/**
 * Method name:getListAnnounce
 * return_tyep:返回一个字符串数组，用于存储列表中所有的 announce
 */
public String[] getListAnnounce() {
    //初始化一个 String 类型的字符数组，用来存储一个或多个 announce 信息
    String as[] = new String[this.getAnnounceList().size()];
    //通过 listiterator 来遍历 announce-list 列表
    for (ListIterator listiterator = this.getAnnounceList().
    listIterator(); listiterator.hasNext();) {
        List list = (List) listiterator.next();
        String s = "";
        //再对 list 进行遍历
        for (Iterator it = list.iterator(); it.hasNext();) {
            //将取出的字节数组转换为字符串
            String s1 = new String((byte[]) it.next());
            if (s != "")
                //在得到的字符串中加入路径分隔符
                s = s + Separator;
            s = s + s1;
        }
        //将得到的 s 依次赋给字符串数组
        as[listiterator.previousIndex()] = s;
    }
    //将得到的 announce-list 信息，以字符串数组的形式返回
    return as;
}
```

以下是处理 info_hash 的方法，因为在访问 tracker 的时候，HTTP 的请求链接地址需将 info_hash 的值进行转义，这是一个将 info_hash 的每个字节变成 16 进制的操作。代码如下：

```
    /**
     * getInfoHashHex，用来处理 info_hash，通过此方法可以得到 info_hash 的值，返
     * 回一个字符串类型的 info_hash 值
     */
    public String getInfoHashHex() {
        //定义一个字符数组，此数组中包含 16 进制位所需的全部字符
        char ac[] = { '0', '1', '2', '3', '4', '5', '6', '7', '8', '9',
        'a', 'b', 'c', 'd', 'e', 'f' };
        //初始化一个名为 tringbufferStringBuffer 对象，并指定初始大小
        StringBuffer stringbuffer = new StringBuffer(InfoHash.length * 2);
        //对 InfoHash 中的每一个字节进行处理
        for (int i = 0; i < InfoHash.length; i++) {
            byte byte0 = InfoHash[i];
            //将取得的字节与 16 进制的 f0 进行“与”操作，并右移 4 位
            stringbuffer.append(ac[(byte0 & 0xf0) >> 4]);
```

```
                stringbuffer.append(ac[byte0 & 0xf]);
            }
            //将 StringBuffer 转换成字符串并作为结果返回
            return stringbuffer.toString();
        }
        /*
         * 以下都是针对 torrent 类各个属性的存取方法，主要用于各种读取操作，具体的内容请参
        考源代码
         */
        public void setListAnnounce(String[] listAnnounce) {
            this.listAnnounce = listAnnounce;
        }
        ...
        public void setNoAnnounceList(boolean noAnnounceList) {
            this.noAnnounceList = noAnnounceList;
        }
        //存取方法结束
}
```

根据.torrent 文件结构，在文件内部还有一个 info 的字典信息，在这个信息里记录了文件的各种信息，所以，需要定义一个单独的名为 InfoContent 的类来表示。

InfoContent 类位于 btclient_test 工程中的 p2p.btclient.torrent 包下，以下是 InfoContent 类中主要方法的说明，详细完整的实现源代码请参考随书光盘。InfoContent 类的文件位置如下：

【示例源代码：ch13\ch13_code\btclient_test\src\p2p\btclient\torrent\ InfoContent.java】

```
package p2p.btclient.torrent;
/**
 * InfoContent 类，主要用来定义一个 info 字典的内容，根据 BT 协议规范，info 字典里包
含以下结构:
 * piece length 整数，每块的大小
 * pieces   整数，  块数
 * private 整数，如果设置为 1，客户端需从 metafile 文件中的 tracker 得到种子，  如
 果没有设置或设置为零，则可从任何途径获得种子
 *************描述单个文件的时候***********
 * name 字符串，文件名
 * length 整数，文件大小
 * md5sum 可选，文件的 md5 值，  32 个字符
 **************描述多个文件的时候**********
 * name 文件夹的名字
 * files 一个字典组成的列表，  每一项有以下关键字
 * lenght 整数，  文件大小
 * md5sum，  字符串，  可选
 * path 路径，文件夹和文件名的 bencode 编码的列表
 */
public class InfoContent {
    //针对 info 字典的结构，构造以下的属性
    private int pieceLength;    //定义 pieceLength，块大小
    private String pieces;      //定义 pieces，一个长度为 20 的整数倍的字符串。
    private int piecesNum;      //定义分块数，由 pieces 计算得到
    private String name;        //定义 name
    private long fileLength;    //定义 fileLength
   private String md5Sum;       //定义 md5Sum
```

```
    private MultiFile multiFile[]; //定义多文件结构，用来存储所有的多文件内容
    private boolean isSingleFile;   //判断是否是单文件
```

以下是 Java 中存取器的实现方法，通过 Seter 和 Getter 来实现，这里有部分省略，完整的代码请参阅随书光盘第 13 章的源代码部分。

```
    /*
     * 以下是这些属性的 getXXX 和 setXXX 的方法，用于存取各个属性值
     */
    public long getFileLength() {                     //取得文件的长度
        return fileLength;
    }
    public void setFileLength(long fileLength) {     //设置文件的长度
        this.fileLength = fileLength;
    }
    public String getMd5Sum() {                      //取得 MD5 的校验值
        return md5Sum;
    }
    public void setMd5Sum(String md5Sum) {           //设置 MD5 的校验值
        this.md5Sum = md5Sum;
    }
    ...                                  //其他的是与之类似的方法，请参考源码
    /**
     * Method name:getPiecesNum
     * TODO : 此方法用于计算文件的分块数，由 Pieces 计算得到，
     * return_type:int
     */
    public int getPiecesNum() {
        return this.getPieces().getBytes().length / 20 ;
    }
}
```

在 info 字典结构里，对单文件和多文件也有不同的区分，对于一个多文件信息而言，它是由多个单文件组成的，所以这里还需要一个描述多文件结构的类，这个类就定义为 MultiFile.java。

MultiFile 类位于 btclient_test 工程中的 p2p.btclient.torrent 包下。以下是 MultiFile 类中主要方法的说明，详细完整的实现源代码请读者参考随书光盘，MultiFile 类的文件位置如下：

【示例源代码：ch13\ch13_code\btclient_test\src\p2p\btclient\torrent\MultiFile.java】

MultiFile 类，主要用来描述.torrent 文件中多文件的信息。关于此类编程步骤和主要的代码说明如下：

```
package p2p.btclient.torrent;
/**
 * MultiFile 类，用于定义一个多文件的结构，根据 BT 协议规范，多文件结果里每一个文件的
 结构是：
 * lenght 整数， 文件大小
 * md5sum， 字符串， 可选
 * path 路径，文件夹和文件名的 bencode 编码的列表
*/
public class MultiFile {
    private long singleFileLength;          //定义多文件里每个文件的长度
    private String md5sum;                  //定义 md5sum
```

```
    private String path;                        //定义 path
    /*
     * 以下是这些属性的 getXXX 和 setXXX 的方法，用于存取各个属性值
     */
    public long getSingleFileLength() {         //取得单个文件的长度
        return singleFileLength;
    }
    public void setSingleFileLength(long singleFileLength) {
                                                //设定单个文件的长度
        this.singleFileLength = singleFileLength;
    }
    public String getMd5sum() {                 //取得 MD5 校验值
        return md5sum;
    }
    public void setMd5sum(String md5sum) {  //设定 MD5 校验值
        this.md5sum = md5sum;
    }
    public String getPath() {                   //取得路径信息
        return path;
    }
    public void setPath(String path) {          //设定路径信息
        this.path = path;
    }
}
```

以上的 3 个类，它们共同完成对.torrent 文件结构的建模，并根据 BT 协议规范定义相应的数据结构。整个.torrent 文件及文件中的信息结构都抽象成一个个具体的类，但要完成整个.torrent 文件的解析过程，还需要一个执行具体解析算法的类。此类命名为 BCode.java，它需要根据 B 编码的规则要求来实现，从字节的角度来文件内容进行解析。

BCode 类位于 btclient_test 工程中的 p2p.btclient.torrent 包下。以下是 BCode 类中主要方法的说明，详细完整的实现源代码请读者参考随书光盘，BCode 类的文件位置如下：

【示例源代码：ch13\ch13_code\btclient_test\src\p2p\btclient\torrent\BCode.java】

BCode 类，是对 BT 协议规范中 B 编码规则的算法实现，包括编码和解码两个部分。其中，解码的大致流程是，先以字节流的形式扫描读取.torrent 文件的内容，再对读取的字节内容根据 B 编码的 4 种数据编码规则进行处理，分别将其中的字典类型、列表类型、整数类型、字符串类型抽取出来，再对抽取的数据进行一系列判定和转换，最后解析出所需的数据值。而编码则是解码的逆过程，详细的实现请参考如下的代码说明。

BCode 类中，即有编码也有解码，所以首先对编码与解码的两个方法进行声明：

```
//定义一个名为 BCode，用来实现.torrent 文件的编码与解码
public class BCode {
   //一个私有的空的构造方法，在此类以外，无法进行实例化
    private BCode() {
    }
   // Bdecode()方法，用来实现.torrent 文件的解码，传入一个字节数组作为参数
    public static Object BDecode(byte abyte0[]) {
       //通过 ByteArrayInputStream 对象，将传入的参数封装成字节数组输入流
        ByteArrayInputStream bytearrayinputstream = new
                                  ByteArrayInputStream (abyte0);
       //调用一个递归的 BdecodeRecursive()解码方法，将数据流进行解码
        Object obj = BDecodeRecursive(bytearrayinputstream);
```

```
        //如果在读取要解码的数据流时没有正常结束，则抛出异常
        if (bytearrayinputstream.read() != -1)
            throw new IllegalArgumentException("不合法的字节流");
        //返回经过解码后的 Object 对象
        else
            return obj;
    }
    // Bencode()方法，用来实现对一个对象的编码，编码成 B 编码结构的字节数组
    public static byte[] BEncode(Object obj) {
        //初始化一个 ByteArrayOutputStream，用于输出字节数组数据流
        ByteArrayOutputStream bytearrayoutputstream = new ByteArrayOut-
        putStream();
        //调用一个递归的 BEncodeRecursive 编码方法，将数据对象编成 B 编码结构
        BEncodeRecursive(obj, bytearrayoutputstream);
        //返回编码后的字节数组
        return bytearrayoutputstream.toByteArray();
    }
```

以上的两个方法，就是 BCode 类中的一个解码方法和一个编码方法，在 Bdecode()方法中，需要传入一个字节数组进行解码，返回一个解码成功的数据对象。而 Bencode()方法，则需要传入一个需要编码的对象为参数，返回的是一个 B 编码结构的字节数组。这也正说明了对.torrent 文件进行编码与解码是一个互逆的过程。

根据 BT 协议规范，B 编码的规则表明了整个 B 编码文件就是一个大的字典结构，在这个大的结构里，又一层层嵌套着其他的字典结构。针对这种数据结构，很适合用递归的方法进行解决，这也是在上述的方法中调用的两个递归方法的原因。下面就说一下这两个递归方法的具体实现。

```
//静态的 BdecodeRecursive()方法，递归地将一个 B 编码的字节数组输入流按编程规则进行
解码
    static Object BDecodeRecursive(ByteArrayInputStream bytearrayinput-
    stream) {
        //对输入流进行判断，是否支持 mark/reset，如果不支持就抛出异常
        if (!bytearrayinputstream.markSupported())
            throw new IllegalArgumentException(bytearrayinputstream.
            getClass()+ " 字节输入流出错");
        //按参数标识的位置，对 bytearrayinputstream 数据流进行标记
        bytearrayinputstream.mark(2);
        //读取一个字节，存储到 i 到中
        int i = bytearrayinputstream.read();
        //调用 reset()方法，将缓冲区的位置重置为标记位置
        bytearrayinputstream.reset();
        //对读取的字符进行判断
        switch (i) {
        //整数 105，表示的是字符 i 的 ASCII 码
        case 105:
            //如果此字符以 i 开头，则调用解码整数数据类型的方法
            return decodeInt(bytearrayinputstream);
        //整数 108，表示字符 l 的 ASCII 码
        case 108:
            //如果此字符以 l 开头，则调用解码 List 数据类型的方法
            return decodeList(bytearrayinputstream);
        //整数 100，表示字符 d 的 ASCII 码
        case 100:
```

```
            //如果此字符以 d 开头，则调用解码字典数据类型的方法
            return decodeMap(bytearrayinputstream);
        }
        //调用 decodeString()方法，解码字符串数据
        return decodeString(bytearrayinputstream);
    }
   //静态的 BencodeRecursive()方法，递归地将一个数据对象，编码成 B 编码规则的数据
    结构
    static void BEncodeRecursive(Object obj,
            ByteArrayOutputStream bytearrayoutputstream) {
        //对数据对象进行判断，属于何种类型的数据，对其中的整数类型数据进行处理
        if ((obj instanceof Number) && !(obj instanceof Float)
                && !(obj instanceof Double) && !(obj instanceof BigDecimal)) {
            //105 表示字符 i 的意思，在字节流中写入字符 i
            bytearrayoutputstream.write(105);
            //将要写入的数据对象转换成字节数组
            byte abyte0[] = obj.toString().getBytes();
            //将数据内容，写入到 bytearrayoutputstream 流中
            bytearrayoutputstream.write(abyte0, 0, abyte0.length);
            //写入完成后加入结束标记 e,101 是字符 e 的 ASCII 码
            bytearrayoutputstream.write(101);
        //对字符串类型的数据进行编码
        } else if (obj instanceof String)
            //递归调用
            BEncodeRecursive(((String) obj).getBytes(), bytearrayoutputs-
            tream);
        //对字节数组进行编码
        else if (obj instanceof byte[]) {
            //将数据对象强制转换
            byte abyte1[] = (byte[]) obj;
            //取得数据长度
            byte abyte2[] = Integer.toString(abyte1.length).getBytes();
            //写入表示长度的数据
            bytearrayoutputstream.write(abyte2, 0, abyte2.length);
            //写入冒号字符，字符“:”的 ASCII 码是 58
            bytearrayoutputstream.write(58);
            //后面接着写入要编码的数据内容
            bytearrayoutputstream.write(abyte1, 0, abyte1.length);
        //对 List 类型的数据进行编码
        } else if (obj instanceof List) {
            //写入 List 类型数据编码的开始字符 l
            bytearrayoutputstream.write(108);
            //对 List 中的数据进行遍历，并编码
            for (Iterator iterator = ((List) obj).iterator(); iterator
                    .hasNext(); BEncodeRecursive(iterator.next(),
                    bytearrayoutputstream));
            //写入完成后加入结束标记 e,101 是字符 e 的 ASCII 码
            bytearrayoutputstream.write(101);
        //对字典类型的数据进行编码
        } else if (obj instanceof Map) {
            //写入字典类型数据编码的开始标记符 d
            bytearrayoutputstream.write(100);
            //将字典类型（Map）类型中所有的 Key 值取出，放到一个字符串数组里
            String as[] = (String[]) ((Map) obj).keySet()
```

```
                .toArray(new String[0]);
        //对 Key 值进行排序
        Arrays.sort(as);
        int i = 0;
        //递归调用 BencodeRecursive()方法，对字典类型数据进行编码
        for (int j = as.length; i < j; i++) {
            BEncodeRecursive(as[i], bytearrayoutputstream);
            BEncodeRecursive(((Map) obj).get(as[i]), bytearrayoutput-
            stream);
        }
      //一条数据编码结束后加入结束标记 e,101 是字符 e 的 ASCII 码
        bytearrayoutputstream.write(101);
        //捕获并处理异常
    } else {
        throw new IllegalArgumentException("错误: "+ obj.getClass());
    }
}
```

在以上的方法中，分别需要对 B 编码规则中规定的 4 种数据类型进行处理。以下几个方法就是对整数型、列表型、字符串类型及字典类型数据的具体处理过程。

```
//对整数类型的数据进行解码
static Number decodeInt(ByteArrayInputStream bytearrayinputstream) {
    int i = bytearrayinputstream.read();
   //判断整数类型的数据是否以字符 i 开始
    if (i != 105)
        throw new IllegalArgumentException("整数编码错误");
    String s;
    //读取字符串，直到结束标记 e 出现，或是文件读取结束
    for (s = ""; (i = bytearrayinputstream.read()) != 101 && i != -1;
    s = s   + (char) i);
    if (i == -1)
        //如果没出现结束标记 e 而文件结束，证明文件不完整，非正常结束
        throw new IllegalArgumentException("非正常结束");
    if (s.length() == 0)
        //如果编码内容的长度为 0，说明整数编码内容为空
        throw new IllegalArgumentException("整数编码内容为空");
    else
        return new Long(s);
}
//对列表（List）类型的数据进行解码
static List decodeList(ByteArrayInputStream bytearrayinputstream) {
    int i = bytearrayinputstream.read();
   //判断列表类型的数据是否以字符 l 开始，否则抛出错误
    if (i != 108)
        throw new IllegalArgumentException("列表编码错误");
   //新建一个 ArrayList 对象
    ArrayList arraylist = new ArrayList();
    //对数据应该按参数设定的值进行标记
    bytearrayinputstream.mark(2);
    //遍历并读取数据流中的数据
    while ((i = bytearrayinputstream.read()) != 101 && i != -1) {
        bytearrayinputstream.reset();
        //调用 BdecodeRecursive()方法，递归地对列表数据编码
        arraylist.add(BDecodeRecursive(bytearrayinputstream));
```

```
            bytearrayinputstream.mark(2);
        }
        //判断文件是否正常结束
        if (i == -1)
            throw new IllegalArgumentException("列表内容非正常结束");
        else
            return arraylist;
    }
    //对字典（Map）类型的数据进行解码
    static Map decodeMap(ByteArrayInputStream bytearrayinputstream) {
        int i = bytearrayinputstream.read();
       //判断字典类型的数据是否以字符 d 开始，否则抛出错误
        if (i != 100)
            throw new IllegalArgumentException("字典编码错误");
        //新建一个 Hashtable 对象
        Hashtable hashtable = new Hashtable();
        String s = null;
        bytearrayinputstream.mark(2);
        //遍历并读取数据流中的数据
        while ((i = bytearrayinputstream.read()) != 101 && i != -1) {
            bytearrayinputstream.reset();
          //对字典内部的字符串数据还需要进行编码
            String s1 = new String(decodeString(bytearrayinputstream));
            if (s != null && s.compareTo(s1) >= 0)
                throw new IllegalArgumentException("错误");
            //调用 BdecodeRecursive()方法，递归的对字典型数据编码
            Object obj = BDecodeRecursive(bytearrayinputstream);
            //将编码后的数据，存入到 Hashtable 中
            hashtable.put(s1, obj);
            bytearrayinputstream.mark(2);
        }
        //判断文件是否正常结束
        if (i == -1)
            throw new IllegalArgumentException("字典编码非正常结束");
        else
            return hashtable;
    }
    //对字符串（String）类型的数据进行解码，返回字节数组
    static byte[] decodeString(ByteArrayInputStream bytearrayinputstream) {
        int i;
        String s;
       //遍历字符串，直到读到冒号“：”、或文件结束
        for (s = ""; (i = bytearrayinputstream.read()) != 58 && i != -1; s
        = s + (char) i);
        //如果没有出现冒号，证明编码不符合规则
        if (i == -1)
            throw new IllegalArgumentException("字符串解析错误");
        //如查字符串长度为 0，说明编码内容不存在
        if (s.length() == 0)
            throw new IllegalArgumentException("字符串内容不存在");
        //将字符串类型的数据转换成整型
        int j = Integer.parseInt(s);
        byte abyte0[] = new byte[j];
        if (bytearrayinputstream.read(abyte0, 0, j) != j)
```

```
            throw new IllegalArgumentException("");
        else
            return abyte0;
    }
```

BCode 类主要是根据 B 编码规则而编写的关于针对.torrent 文件的编码和解码的实现，此类或者是与此类似的 B 编码类，网络上有很多现成的源码，只要满足 B 编码规则的算法，都可以实现 BCode 类的功能。读者也可以自己尝试写一个.torrent 文件的编码与解码类。

在实际的系统中，调用 BCode 类就可以完成.torrent 文件的制作和解析，下面就会具体地讲解一下，如何通过这个类解析一个.torrent 文件。

13.4.3　.torrent 文件解析结果

有了以上的解析过程，就可以编写相应的测试程序来测试一下对.torrent 文件解析的结果了。所以编写一个测试类，名为 GetTorrentInfo.java，此类完成对 torrent 的文件的解析，并将此文件中包含的主要信息打印出来。

GetTorrentInfo 类位于 btclient_test 工程中的 p2p.btclient 包下。以下是 GetTorrentInfo 类中主要方法的说明，详细完整的实现源代码请读者参考随书光盘，GetTorrentInfo 类的文件位置如下：

【示例源代码：ch13\ch13_code\btclient_test\src\p2p\btclient\ GetTorrentInfo.java】

```
package p2p.btclient;
import java.io.*;
import java.util.*;
import p2p.btclient.torrent.*;
import p2p.btclient.tracker.ConnectTracker;
public class GetTorrentInfo {
    // 指定 torrent 文件所在的位置
    static File torrent = new File("D:/project/btclient test/torrent/test.
    torrent");
    public static void main(String[] args) {
        try {
            //根据输入流通过 FetchTorrentInfo 类，取出 torren 文件的所有内容
            Torrent ft = new Torrent(torrent);
            //info 字典结构的内容，里面描述了文件的一些信息， 包括两种情况，一是单个
            文件，另一个是包括目录的多个文件
            InfoContent tt = new InfoContent();
            /*
             * announce 在 torrent 文件中有两种形式，只有一个 announce 时，它是以字
            符串形式存储的如果有两个以上的 announce 的时候，它就是以列表形式存储的
             */
            System.out.println("Torrent 文件全部信息如下：");
            System.out.println("------------Torrent 文件 Announce 信息
            -------------------");
            if(ft.isNoAnnounceList()) {
                //只有一个 Announce 时，取出此值
                System.out.println("announce =" + ft.getAnnounce());
            }else {
                String[] str = ft.getListAnnounce();
                //如果是 announce 列表，则取每一个 announce 值
```

```
                for (int i = 0; i < str.length; i++) {
                    String announcelist = str[i];
                    System.out.println("announcelist " + i + "=" +
                    announcelist);
                }
            }
            System.out.println();
            System.out.println("-------Torrent 文件基本信息------------");
            System.out.println("getCreationDate=" + ft.getCreation
            Date());
            System.out.println("getComment=" + ft.getComment());
            System.out.println("getCreatedBy=" + ft.getCreatedBy());
            System.out.println("getInfoHashHex=" + ft.getInfoHashHex());
          System.out.println();
            System.out.println("------torrent 文件 info 字典信息---------");
            tt = ft.getTorrentInfo();
            if( tt.isSingleFile()){
                System.out.println("此文件是单文件");
            }else {
                System.out.println("此文件是多文件");
            }
            System.out.println("文件分块数：" + tt.getPiecesNum());
            if (tt.isSingleFile()) {
                System.out.println("----torrent 的单文件信息如下：-------");
                System.out.println("文件名" + tt.getName() + '\t' + "文件大
                小" + tt.getFileLength() + '\t' + "分块长度" + tt.getPieceL
                ength());
            } else {
                System.out.println("------torrent 的多文件信息如下:------");
                System.out.println("文件名:" + tt.getName() + '\t' + '\t' +
                "文件大小:" +  tt.getFileLength() );
                MultiFile[] temp = tt.getMultiFile();
                System.out.println("多文件列表信息：");
                for (int i = 0; i < temp.length; i++) {
                    String path = temp[i].getPath();
                    long length = temp[i].getSingleFileLength();
                    System.out.println("文件名:" + path  + '\t' + '\t' + "
                    文件大小:" + length);
                }
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

由程序代码可知，此测试程序要解析的.torrent 文件位于 D:/project/btclient_test/torrent/路径下，名为 test 的.torrent 文件，解析文件内容结果如图 13.4 所示。

图 13.4 所示的是从 Eclipse 控制台中截取的打印信息，这些打印信息清楚展示了 test.torrent 文件的主要信息内容。为了对比，图 13.5 所示的是用 BitComet 打开此 test.torrent 文件的结果示意图。

注意：BitComet，是一款 BT 客户端软件，主要用于进行 BT 下载。

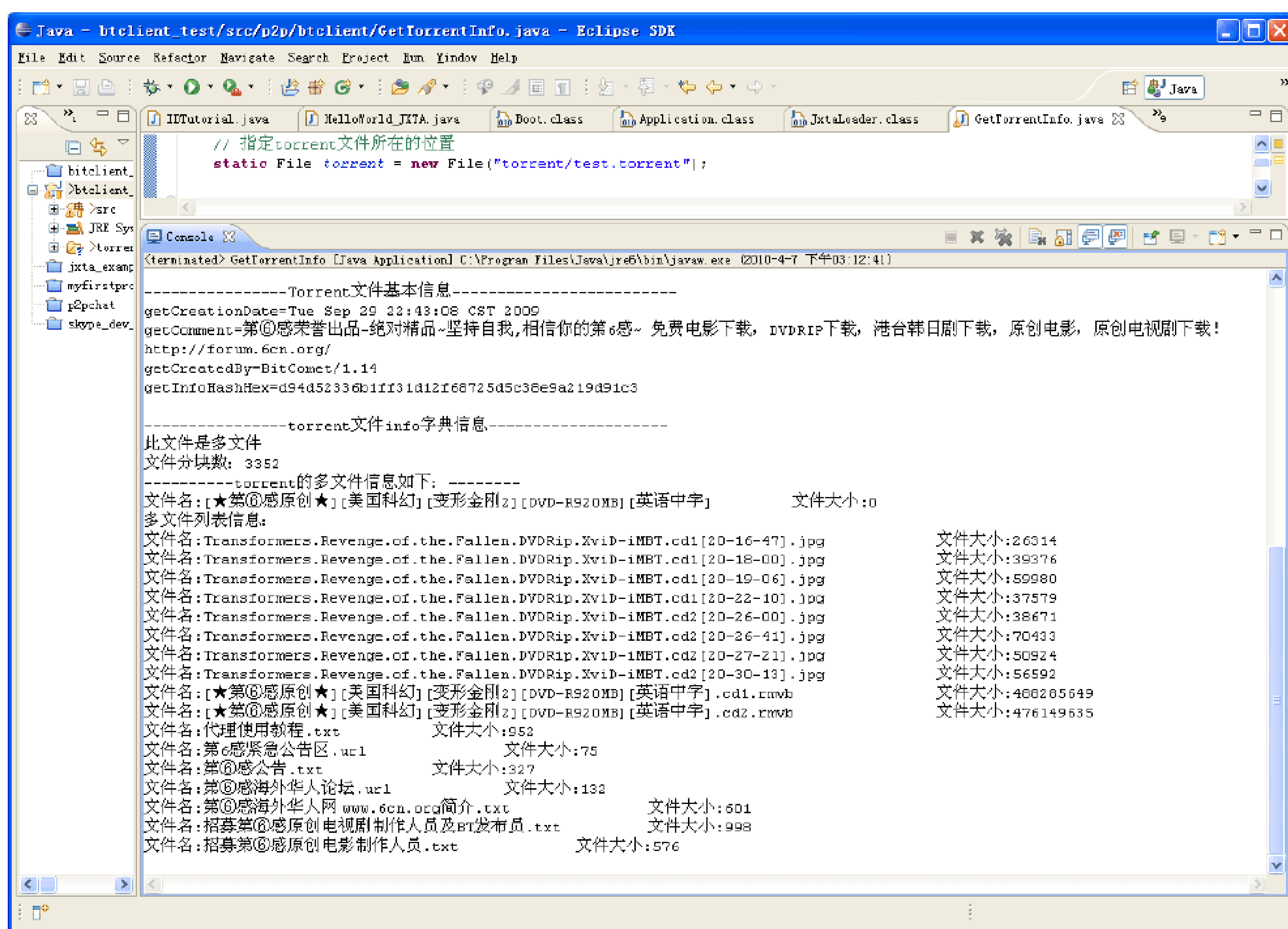


图 13.4　程序解析.torrent 文件后的打印结构

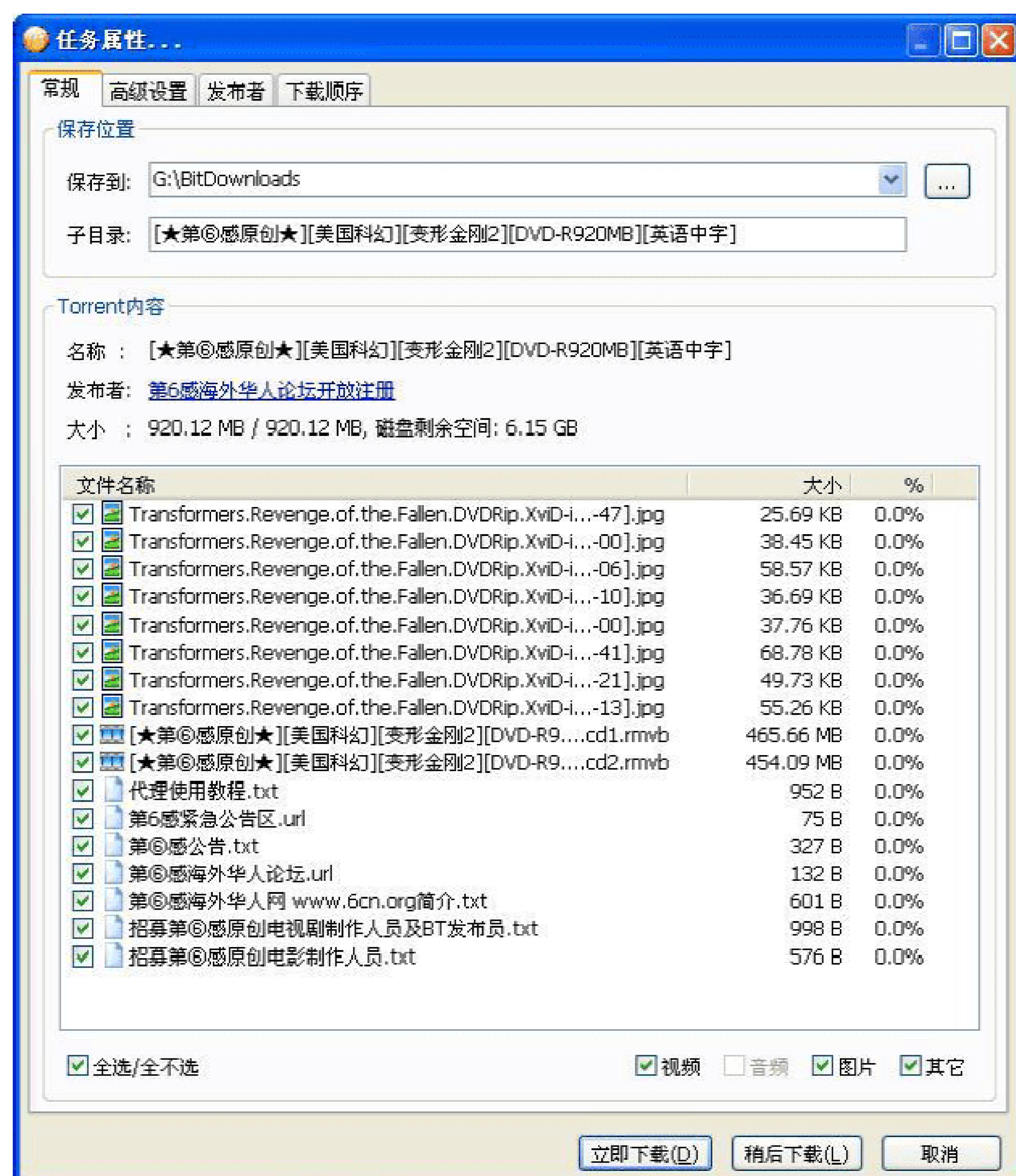


图 13.5　BT 客户端软件解析 test.torrent 文件的结果

由图 13.4 与图 13.5 的对比，可以清楚地看出，由程序解码.torrent 文件得到的内容信息与 BT 客户端得到的内容是完全一样的。这就说明，通过以上几个程序，test.torrent 文件可以被成功地解析出来。

13.5　与 Tracker 服务器交互得到 Peer 信息

Tracker 是一个响应 HTTP GET 请求的 HTTP/HTTPS 服务。这个请求包含来自客户端的度量信息，这些信息能够帮助 Tracker 全面地统计 torrent。Tracker 的响应包含一个 peers 列表，这个列表能够帮助客户端加入到.torrent 文件中。

13.5.1　Peer 发向 Tracker 服务器的请求

Peer 结点向 Tracker 服务器发送的请求信息是通过一个 URL 来表示的，这个 URL 就是 Base URL，它由.torrent 文件中定义的 announce URL 组成，然后使用标准的 CGI 方式将这些请求参数追加到这个 URL 后面(CGI 方式即在 announce URL 后面紧跟一个‘?’，然后是一个以‘&’分隔的‘param=value’序列)。此 URL 请求的参数是由 BT 协议规范定义的，主要参数说明如下。

- ❑ info - hash：元信息文件中 info 关键字/值的 20 个字符的 SHA1 散列值。特别地，由于必须向 Tracker 服务器传递二进制的 info hash 值，因此 info_hash 参数的值需符合 HTTP 协议中对二进制数据的传输规定，转化为%nn 的形式。
- ❑ peer- id：标识结点唯一的 20 个字节的字符串。
- ❑ port：结点监听的端口号，peer 所监听的端口。下载者通常在 6881 端口上监听，如果该端口被占用，那么会一直尝试到 6889；如果都被占用，那么就放弃监听。
- ❑ up loaded：上传的字节数。
- ❑ downloaded：下载的字节数。
- ❑ left：还需要下载的字节数，作为一个新的结点，这里就是所有文件的总长度。
- ❑ event：取值必须是 started、comp leted、stoppe 其中之一，或者为空，等同于未使用。
- ❑ ip：指明客户端机器的真实 IP 地址。
- ❑ numwant：可选参数，指明结点希望从追踪服务器得到的其他结点个数。默认值通常是 50。

以上给出了与 Tracker 服务器进行交互所需要的全部参数，然而在实际的交互过程中并非所有参数都需要填写，只要交互参数满足你的查询请求即可，就可以提交给 Tracker 服务器进行交互。以下就是一个标准的 HTTP 的 GET 请求。

http://TrackerURL?info_hash=测量参数中的 info hash 值&peer_id=任何 20 字节的二进制数据&port=6881－6889 之间的端口号&uploaded=0&downloaded=0&left=所有文件总长度&compact=1&event=started&numwant=65535。

在这个请求的 URL 中，其中 Tracker URL 是种子文件中给出的 Tracker 服务器的链接地址，info hash 为用户指定测量的种子索引值。还有 peer_id、left 等信息都可以通过解析.torrent 文件得到，如图 13.6 所示，就是通过.torrent 文件解析程序解析 test.torrent 文件后打印出的 announce URL 列表。

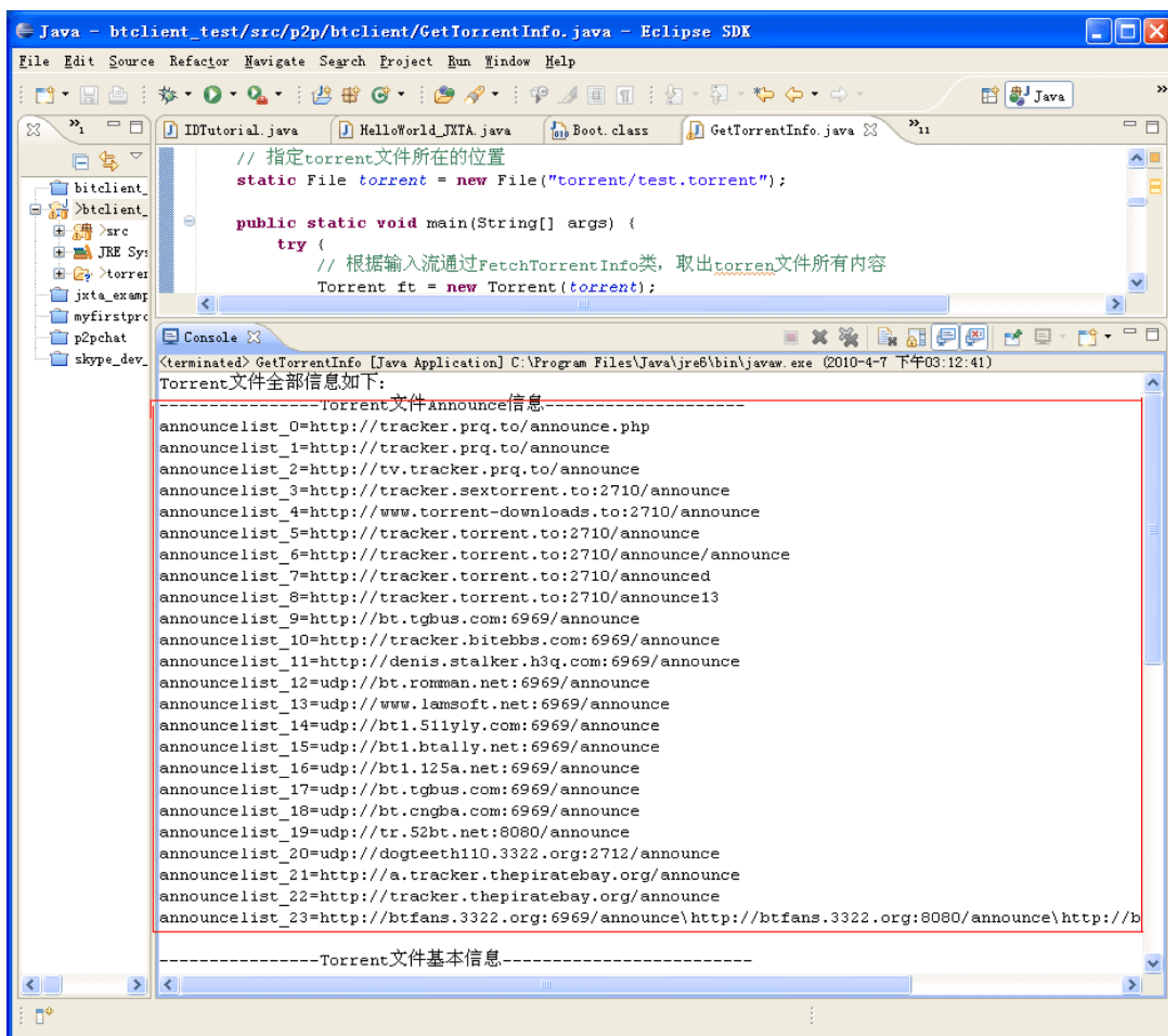


图 13.6　.torrent 文件解析程序解析 test.torrent 文件后得到的 Tracker 列表

13.5.2　连接 Tracker 服务器

有了以上这些信息，就可以通过发送 URL 的 GET 请求来连接 Tracker 服务器了。可以编写一个 ConnectTracker 类来实现 Peer 结点与 Tracker 服务器的连接。

ConnectTracker 类位于 btclient_test 工程中的 p2p.btclient.tracker 包下。以下是 ConnectTracker 类中主要方法的说明，详细完整的实现源代码请读者参考随书光盘，ConnectTracker 类的文件位置如下：

【示例源代码：ch13\ch13_code\btclient_test\src\p2p\btclient\tracker\ConnectTracker.java】

```
/**
 * ConnectTracker.java，通过发送一个 Base URL 来连接 Tracker 服务器，并得到服务器
 * 的响应消息。编程步骤及主要的代码说明如下：
 */
package p2p.btclient.tracker;
import java.io.*;
import java.net.MalformedURLException;
```

```
import java.net.*;
import java.util.*;
import java.util.zip.GZIPInputStream;
import p2p.btclient.torrent.BCode;
import p2p.btclient.torrent.*;
/**
* ConnectTracker 类，主要用于连接 Tracker 服务器的操作，并从 Tracker 里得到 Peer 列
表的信息
* 本类主要有两个方法，一个方法是查询 Tracker 服务器，另一个方法用于得到 Tracker 的响应
结果
*/
public class ConnectTracker {
    @SuppressWarnings("unchecked")
    private List peerList;
    long interval;
    /**
     * Method name:queryTracker TODO :
     * 用于查询 Tracker 服务器，接收一个 File 类型的 torrent 文件参数，返回一个 set 集
    合，用于存储 Peer 结点信息
     * return_type:Set<String>，将得到的 Peer 信息都放到一个 Set 集合里
     */
    public Set<String> queryTracker(File file) throws IOException {
        Set peerSet = new TreeSet();          // 定义一个 TreeSet 集合
        Torrent torrent = null;               // 定义一个 torrent 文件结构
        long filelength = 0;                  // 定义文件的长度
        String peerStr = "";                  // 定义一个表示 Peer 信息的字符串
        Map temmap = null;                    // 定义一个临时的 Map 结构
        // 定义一个临时的 List 结构
        List<String> templist = new ArrayList<String>();
        // 定义存储 announce_list 的列表
        List<String> announcelist = new ArrayList<String>();
        try {
            // 调用 Torrent()方法，解析此文件
            torrent = new Torrent(file);
        } catch (IOException e) {
            e.printStackTrace();
        }
        if(torrent.isNoAnnounceList()){
            announcelist.add(torrent.getAnnounce());
        }else {
            // 得到所有的 announce 列表，用于组建请求 Tracker 的 URL
            String[] str = torrent.getListAnnounce();
            for (int i = 0; i < str.length; i++) {
                announcelist.add(str[i]);
            }
        }
        // 取得 info_hash 值，用于组建请求 Tracker 的 URL
        String hash = "&info_hash="
                + BencodeUtils.escapeString(torrent.getInfoHashHex());
        // 取得文件长度，用于组建请求 Tracker 的 URL，如果是单文件，则就是文件的长度，
        如果是多文件，则是每个文件长度的和
        if (torrent.getTorrentInfo().isSingleFile()) {
            filelength = torrent.getTorrentInfo().getFileLength();
        } else {
            MultiFile[] temp = torrent.getTorrentInfo().getMultiFile();
            for (int i = 0; i < temp.length; i++) {
                // 多文件的时候，所有文件的长度和
```

```
                filelength = filelength + temp[i].getSingleFileLength();
            }
        }
        announcelist = BencodeUtils.parseAnnounce(announcelist);
        // 对每个 annouce 都发出 Tracker 的查询请求，也就是说，所有的服务器都请求
        一遍
        for (Iterator it = announcelist.iterator(); it.hasNext();) {
            String announce = (String) it.next();
            try {
        // 得到 Tracker 的响应信息，Tracker 的响应也是 B 编码格式的数据，还需要进行解
           码操作
                Object obj = BCode.BDecode(ConnectTracker.getTracker-
                Response(file, announce, hash, filelength));
                //如果得到的响应信息，是一个字典结构，解码后强制转换成 Map 类型
                if (obj instanceof Map) {
                    temmap = (Map) BCode.BDecode(ConnectTracker
                            .getTrackerResponse(file, announce, hash,
                            filelength));
                } else {
                    System.out.println("Tracker 服务器返回无法解析的文件！");
                    continue;
                }
            } catch (IOException e) {
                System.out.println("与 Tracket 服务器的连接出现异常，此 anno-
                unce 地址：" + announce+ "无法与服务器连接");
                continue;
            }
            // 解析 Tracker 的响应信息，从中得到 Peer 的信息，是根据 Tracker 的响应
               的结构进行求值的
            if (temmap.containsKey("peers")) {
                try {
                    //得到 Peer 列表信息
                    peerList = (List) temmap.get("peers");
                } catch (ClassCastException e) {
                    System.out.println("得到 Tracker 的响应，但无法解析");
                }
            } else {
                System.out.println("此 announce 地址：" + announce + "的应答
                文件中不含键值 peers");
                continue;
            }
            // "interval"在 BT 协议规范中，表示隔多长时间查询一次 Tracker
            if (temmap.containsKey("interval")) {
                interval = ((Number) temmap.get("interval")).longValue();
            } else {
                System.out
                        .println("此 announce 地址：" + announce + "的应答文
                        件中不含键值 interval");
                continue;
            }
            // 将所有 Peer 的 IP 地址、端口号存放到 PeerSet 里
            for (int i = 0; i < peerList.size(); i++) {
                //取得 peerList 中的第 i 个值，存入 Map 中
                Map map1 = (Map) peerList.get(i);
                //通过值为 Ip 的 Key，取得对应的 Ip 地址值
                byte abyte0[] = (byte[]) map1.get("ip");
                //通过值为 port 的 Key，取得对应的端口值
                long port = ((Number) map1.get("port")).longValue();
```

```
                //将 IP 和对应的端口，连接成字符串
                String tmp = new String(abyte0) + ":" + port;
                if (peerSet.isEmpty() || !peerSet.contains(tmp)) {
                    //将 IP 和端口信息添加到 peerSet 中
                    peerSet.add(tmp);
                }
            }
        }
        return peerSet;
    }
    /**
     * getTrackerResponse()方法，用来得到 Tracker 服务器的响应结果，返回一个字节
    数组
     */
    public static byte[] getTrackerResponse(File file, String announce,
            String infohash, long filelength) throws IOException {
        // 定义请求 Tracker 服务器的 URL 链接结构，此结构在文中有详细的说明
        String queryURL = announce + "?peer id="
                + BencodeUtils.getRandomString(20) + infohash
                + "&port=6885&uploaded=0&downloaded=0&left=" + filelength
                + "0&compact=1&event=started&numwant=5000";
        URL url = null;
       System.out.println("当前查询 Tracker 的 URL: " + queryURL);
        try {
            url = new URL(queryURL);
        } catch (MalformedURLException malformedurlexception) {
            throw new RuntimeException(malformedurlexception);
        }
        Object obj1;
        // 与 Tracker 服务器建立连接
        URLConnection urlconnection = url.openConnection();
        // 设置请求的格式
        urlconnection.setRequestProperty("User-Agent", "TrackPeer v0.1");
        urlconnection.setRequestProperty("Accept-Encoding", "gzip");
        // 得到请求结果
        Object obj = urlconnection.getInputStream();
        obj1 = urlconnection.getHeaderField("Content-Encoding");
        // 处理请求结果，请求结果是 gzip 格式的压缩包，还需要进行解包操作
        if (obj1 != null && ((String) (obj1)).equals("gzip"))
            obj = new GZIPInputStream(((InputStream) (obj)));
        obj1 = new ByteArrayOutputStream();
        int i;
        // 读取请求结果，也是 B 编码格式的结果
        while ((i = ((InputStream) (obj)).read()) != -1)
            ((ByteArrayOutputStream) (obj1)).write(i);
        return ((ByteArrayOutputStream) (obj1)).toByteArray();
    }
}
```

13.5.3　得到 Tracker 服务器的响应消息

Tracker 的响应信息也是用 bencoded 编码的字典表示的，如果 tracker 的响应中有一个关键字 failure reason，那么它对应的是一个字符串，用来解释查询失败的原因；其他关键字都不再需要了，否则，它必须有两个关键字：

- Interval：下载者在两次发送请求之间的时间间隔。
- Peers:一个字典的列表，每个字典包括关键字 peer id、Ip、Port，分别对应 peer 所选择的 ID、IP 地址或者 dns 名称、端口号。

注意： 如果某些事件发生，或者需要更多的 peers，那么下载者可能不定期地发送请求。总地来说，接收到 Tracker 的响应消息就可以得到其他的 Peer 结点的信息，有了这些信息就可以实现 Peer 结点之间的通信。在 ConnectTracker 类中，有一个 getTrackerResponse()方法，这个方法就是得到 Tracker 的响应结果，具体的实现方法读者可参考相应的源代码。

要得到 Peer 结点的信息，还需要定义一个 Peer 类，为了简单地说明问题，这里只取 Peer 的 IP 地址和端口这两个属性，Peer 类位于 btclient_test 工程中的 p2p.btclient.peer 包下。以下是 Peer 类中主要方法的说明，详细完整的实现源代码请参考随书光盘，Peer 类的文件位置如下：

【示例源代码：ch13\ch13_code\btclient_test\src\p2p\btclient\peer\Peer.java】

```
package p2p.btclient.peer;
/**
 * Peer 类，主要用来定义一个 Peer 的信息，Peer 的信息这里只取两个，一个是 IP 地址，另一个是端口号
*/
public class peer {
    private String ip;                  // 定义 IP
    private long port;                  // 定义端口
    //针对 IP 和 Port 的存取操作
    public String getIp() {             //取得 IP 地址
        return ip;
    }
    public void setIp(String ip) {      //设定 IP 地址
        this.ip = ip;
    }
    public long getPort() {             //取得端口信息
        return port;
    }
    public void setPort(long port) {    //设定端口的值
        this.port = port;
    }
}
```

通过 ConnectTracker 类和 Peer 类，就可以从 Tracker 服务器的响应消息里得到活跃的 Peer 结点信息。

13.5.4　Peer 结点信息的获取

以上说明了 Peer 结点信息的获取方法和具体的实现代码。下面就可以编写一个测试类来测试一下如何通过一个.torrent 文件来获取关于此文件的所有活跃的 Peer 结点信息。此类命名为 GetPeersInfo.java。

GetPeersInfo 类，位于 btclient_test 工程中的 p2p.btclient 包下，以下是 GetPeersInfo 类中主要方法的说明，详细完整的实现源代码请参考随书光盘。GetPeersInfo 类的文件位置如下：

【示例源代码：ch13\ch13_code\btclient_test\src\p2p\btclient\GetPeersInfo.java】

```
/**
 * 用来获取一个当前.torrent 文件的所有结点信息，将所有活跃结点的 IP 地址、端口号等打印
 * 输出。编程步骤与主要的代码说明如下：
 */
package p2p.btclient;
import java.io.*;
import java.util.*;
import p2p.btclient.torrent.*;
public class GetPeersInfo {
    public static void main(String[] args) {
        //初始化需要引用的类
        Torrent ft = null;
        ConnectTracker pi = new ConnectTracker();
       //指定一个需要解析的.torrent 文件
        File torrent = new File("D:/project/btclient_test/torrent/test.
        torrent");
        //定义一个 TreeSet()用于存储所有的 Peer 信息
        Set resSet = new TreeSet();
        //用于统计当前的 Peer 数
        int peerNum = 0;
        try {
            ft = new Torrent(torrent);
        } catch (IOException e1) {
            e1.printStackTrace();
        }
        /*
         * 先打印出所有 Announce 的地址，在查询 Tracker 时分别对每一个 announce 都发
        现一个查询：
         *announce 在.torrent 文件中有两种形式，只有一个 announce 时，它就是以字符
        串形式存储的
         * 如果有两个以上的 announce 时，它就是以列表形式存储的
         */
        System.out.println("-------Torrent 文件 Announce 信息--------");
        if(ft.isNoAnnounceList()) {
            //只有一个 Announce 时，取出此值
            System.out.println("announce =" + ft.getAnnounce());
        }else {
            String[] str = ft.getListAnnounce();
            // 如果是 announce 列表，则取每一个 announce 值
            for (int i = 0; i < str.length; i++) {
                String announcelist = str[i];
                System.out.println("announcelist_" + i + "=" + announc-
                elist);
            }
        }
        System.out.println("\n 针对以上每个 Announce 都向 Tracker 服务器发送查询
        请求\n");
        //取得 Peer 信息
        try {
            //查询 Tracker，得到 Peer 信息的结果集合
            resSet = pi.queryTracker(torrent);
```

```
                peerNum = resSet.size();
                if(resSet.isEmpty()) {
                    System.out.println("当前没有任何 Peer! ");
                }else {
                    for(Iterator it = resSet.iterator(); it.hasNext();){
                        String res = (String) it.next();
                        System.out.println("当前活跃 Peer, IP: Port="+ res);
                    }
            }
            System.out.println("本次连接共返回个"+ peerNum +"peer");
            } catch (IOException e) {
            }
        }
}
```

由 GetPeersInfo 类的源代码可知，系统接收一个.torrent 文件为参数，然后将当前在交互此文件的所有 Peer 结点都打印输出。传入的文件还是“D:/project/btclient_test/torrent/”目录下名为 test 的.torrent 文件，程序运行后打印输出信息截图如图 13.7 所示。

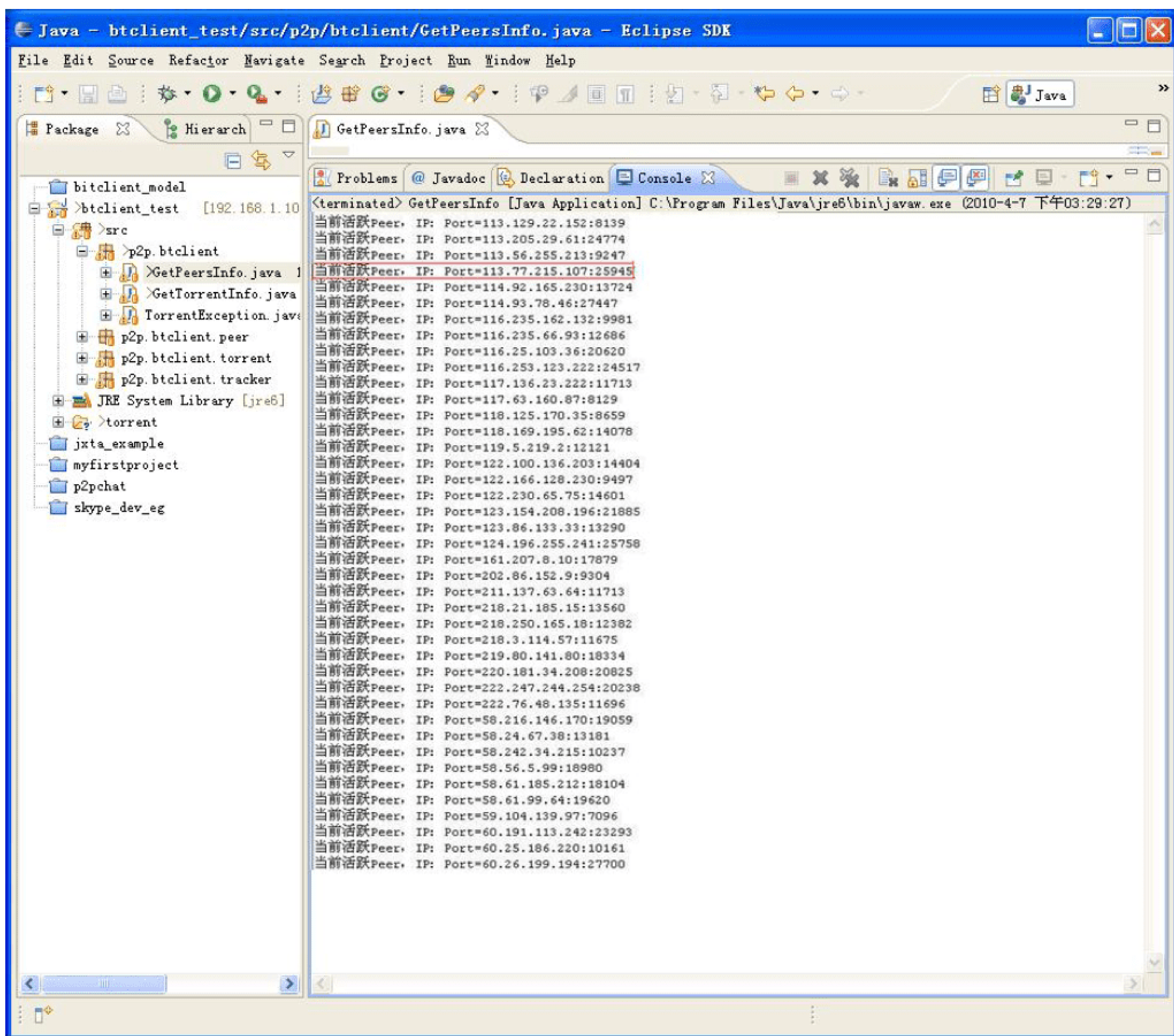


图 13.7　通过询问 Tracker 服务器得到的 Peer 列表信息

图 13.7 所示的是 Eclipse 输出控制台下打印的信息，这只是截取了一部分的列表，还有很多没有显示完全。从图 13.7 中可以清楚地看出当前有哪些 Peer 正在交互 test.torrent 文件中所描述的资源内容。为了验证结果，在 BitComet 中打开 test.torrent 文件后，选择“用户列表”以查看当前活跃 Peer 的情况。如图 13.8 所示的就是针对 test.torrent 种子文件在

BitComet 显示的用户列表情况。

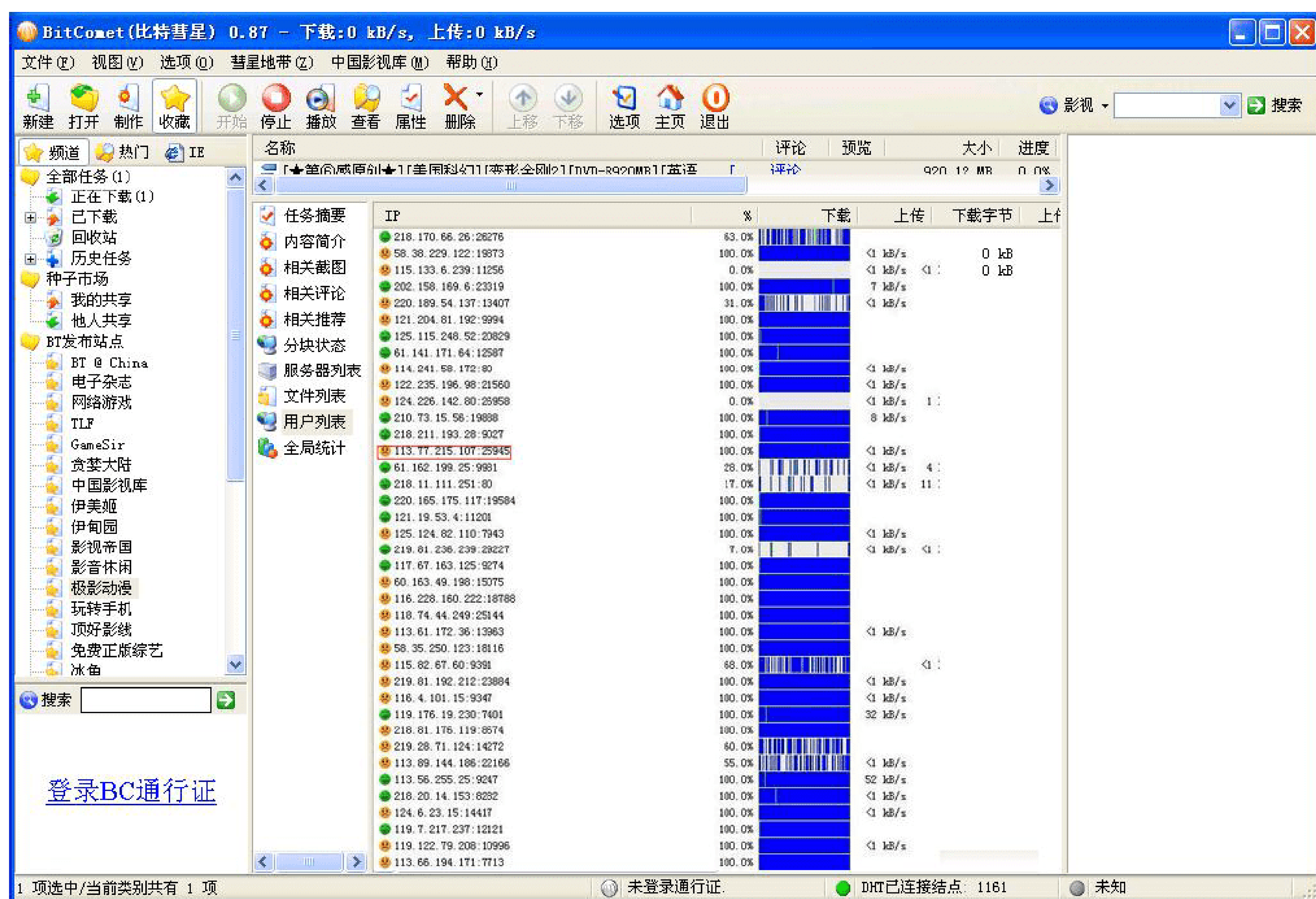


图 13.8　在 BitComet 客户端中 Peer 列表的显示情况

由图 13.7 与图 13.8 对比可以清楚地看出，图 13.8 中框起来作为标记的 IP 地址、端口号为 113.77.215.107:25945 的 Peer 结点信息中，由程序得到的结果与 BitComet 的客户端得到的结果是一样的，证明，程序实现了 Peer 结点发现的功能。

注意： 因为 Peer 结点是可以自由加入和退出的，动态性很强，不同的时刻得到的 Peer 情况是不相同的。而图 13.7 所示由程序得到的 Peer 列表还有很大一部分没有显示出来，所以，完全一样的结点信息并不多。

以上所有程序代码都可以在随书光盘中获取，具体的工程名字在第 13 章对应的目录下名为 bitclient_test 工程中。读者在已经拥有.torrent 文件前提下，可直接将此工程部署到 Eclipse 中，更改.torrent 文件所在路径即可运行。

13.6　Peer 之间交互进行文件下载

在上文的讲解中，已经可以得到 Peer 结点的信息，对 Peer 结点的 IP 地址、端口等也都可以直接打印出来。有了 IP 地址和端口，就可以实现 Peer 间的相互通信了。

13.6.1　BT 对等协议

在 BT 协议规范中，Peer 与 Peer 之间的交互是由 BT 对等协议（peer wire protocol）来

规定的，BT 对等协议基于 TCP 协议的，是 peer 与 peer 之间交换信息的协议。对等的两个 Peer 间的交互协议是对称的，消息在两个方向上同样地传递，数据也可以在任何一个方向上流动。

一旦某个 peer 下载完一个片段，并且检查了它的完整性，那么它就向所有的 peers 宣布拥有了这个片段。连接的任何一端都包含两比特的状态信息，是否阻塞和是否感兴趣。阻塞的时候，是通知对方没有数据可以发送，直到清除阻塞（unchoking）发生时，peer 的数据交互才开始。

一旦一端状态变为 interested，另一端变成非 choking，传输就开始了。也就是说一个 peer 如果想从它的某个 peer 获得数据，会先发送一个消息过去，将连接状态改为 interested，接收到该消息的 peer，先检查是否该给这个 peer 发送数据，如果它对这个 Peer 是阻塞的，那么就可以给它发数据，否则不能发数据。Interested 状态必须一直被设置。

对等协议由一个握手开始，后面是循环的消息流，每个消息前面都有一个数字来表示消息的长度。握手的过程首先是先发送 19，然后发送 BT protocol，19 就是它的长度。后续的所有整数都采用 big-endian 来编码为 4 个字节。

在协议名称之后，是 8 个保留的字节，这些字节当前都设置为 0。

接下来对元文件的 Info 信息进行 SHA 散列，得到 20 字节长的字串。接收消息方，也会对 info 进行 hash 运算，如果结果不一样，说明对方要的文件不是自己提供的，切断连接。接下来是 20 字节长的 peer id。

以上描述的 Peer 之间的握手消息可由表 13.1 直观的表示。

表 13.1　HANDSHAKE消息格式

1 字节	19 字节	1 字节	20 字节	20 字节
19	BT protocol	0	Info hash	peer id

两个 Peer 之间就是通过发送表 13.1 所示的 HANDSHAKE 消息来完成握手过程的。握手完毕。之后是长度固定的交互信息流。零长度信息用来保持连接，被忽略。这种信息一般 2 分钟发出一次，但是在等待数据期间很容易超时。所有非保持连接用信息开头的字节给出类型，可能值如下：

- 0- choke；
- 1- unchoe；
- 2- interested；
- 3- not interested；
- 4-have；
- 5- bitfield；
- 6- request；
- 7-pieces；
- 8-cancel。

在以上的这些消息中，choke、unchoe、interested、not interested 类型的消息没有载荷。

- bitfield 信息仅作为首信息发出。它负载一个比特组，下载者有索引的设为 1，其他为 0。开始下载时没有任何数据的下载者跳过 bitfield 信息。首字节高位到低位对应索引 0～7，依次类推，第二字节对应 8～15 等。尾部剩余的比特位设为 0。

- ❑ have 类型的消息，后面的数据是一个简单的数字，它是下载者刚刚下载完并检查过完整性的片断索引。
- ❑ request 类型的消息，后面包含索引、开始位置和长度，长度是 2 的幂。当前的实现用的是 215，而关闭连接的时候，请求一个超过 2 17 的长度（这种类型的消息，就是当一个 peer 希望另一个 peer 给它提供片断的时候，发出的请求）。
- ❑ cancel 类型的消息，它的数据和 request 消息一样。它们通常只在下载趋向完成的时候发送，也就是在结束模式阶段发送。在一次下载接近完成的时候，最后的几个片断需要很长时间才能下载完。为了确保最后几个片断尽快下载完，它向所有的 peers 发送下载请求。为了保证这样不会带来可怕的低效，一旦某个片断下载完成，它就向其他 peers 发送 cancel 消息。
- ❑ piece 类型的消息，后面保护索引号、开始位置和实际的数据。

注意：这种类型的消息和 request 消息之间有潜在的联系，因为通常有了 request 消息之后，才会响应 piece 消息。如果 choke 和 unchoke 消息发送的过于迅速，或者，传输速度变得很慢，那么可能会读到一些并不是所期望的片断。

以上描述的就是 Peer 间进行交互的协议。关于 BT 的 Peer 端协议，在本书的第 6 章 BT 协议规范分析中也有详细的说明，这里是对以前所学内容的进一步补充和分析，请读者对照着学习。

13.6.2　Peer 间交互的实现方法

关于 Peer 间交互的实现方法，主要就是根据 Peer 端协议的规定进行握手再进行一系列消息交互的过程，这一过程主要通过两个 Peer 结点之间的 Socket 通信，将交互的消息封装成相应的数据包，在彼此之间进行交互通信。在用程序实现过程中，要重点处理好不同类型消息的封装，在文件下载的时候，还要处理好存储空间的分配、文件分块的校验和处理、下载进度的控制等。

关于 Peer 间通信的具体实现，这里就不再讲解。在本书的随书光盘中，提供了一个 BT 客户端简易的实现原型，此客户端运行的效果图如图 13.9 所示。

注意：在随书光盘的第 13 章目录下，有一个名为 bitclient_model 的工程中，直接将其部署到 Eclipse 平台中即可运行。

以上 BT 客户端的实现原型，只是对 BT 协议的一个简单实现，还有很多地方存在问题，不可以达到应用的地步。读者可以在此基础上进行扩展开发，有兴趣的读者可以自己去实现独立的 BT 客户端系统。

综合上文所讲的，解析.torrent 文件的方法得到元文件信息，再根据 Peer 与 Tracker 交互协议得到 Peer 结点信息，有了 IP 地址和端口再根据 Peer 协议实现 Peer 间交互，那么两个 Peer 间的通信一旦完成，Peer 与 Peer 之间的文件交互也就开始了，这样 BT 的 P2P 特性也就实现了。

以上所讲的就是 BT 客户端的基本实现方法，在讲解本章内容的时候，本书提供了两

个 Java 工程作为示例展示给读者，一个是 btclinet_model 工程，此工程是一个完整的 BT 客户端的实现原型，读者可以自行参考相应的源代码，以了解一个完整的 BT 客户端实现方法。另一个是 btclient_test 工程，此工程是对 BT 客户端核心工作过程的抽取和详细说明，要深入理解 BT 的工作原理，就需要认真地研究此工程的实现源代码。

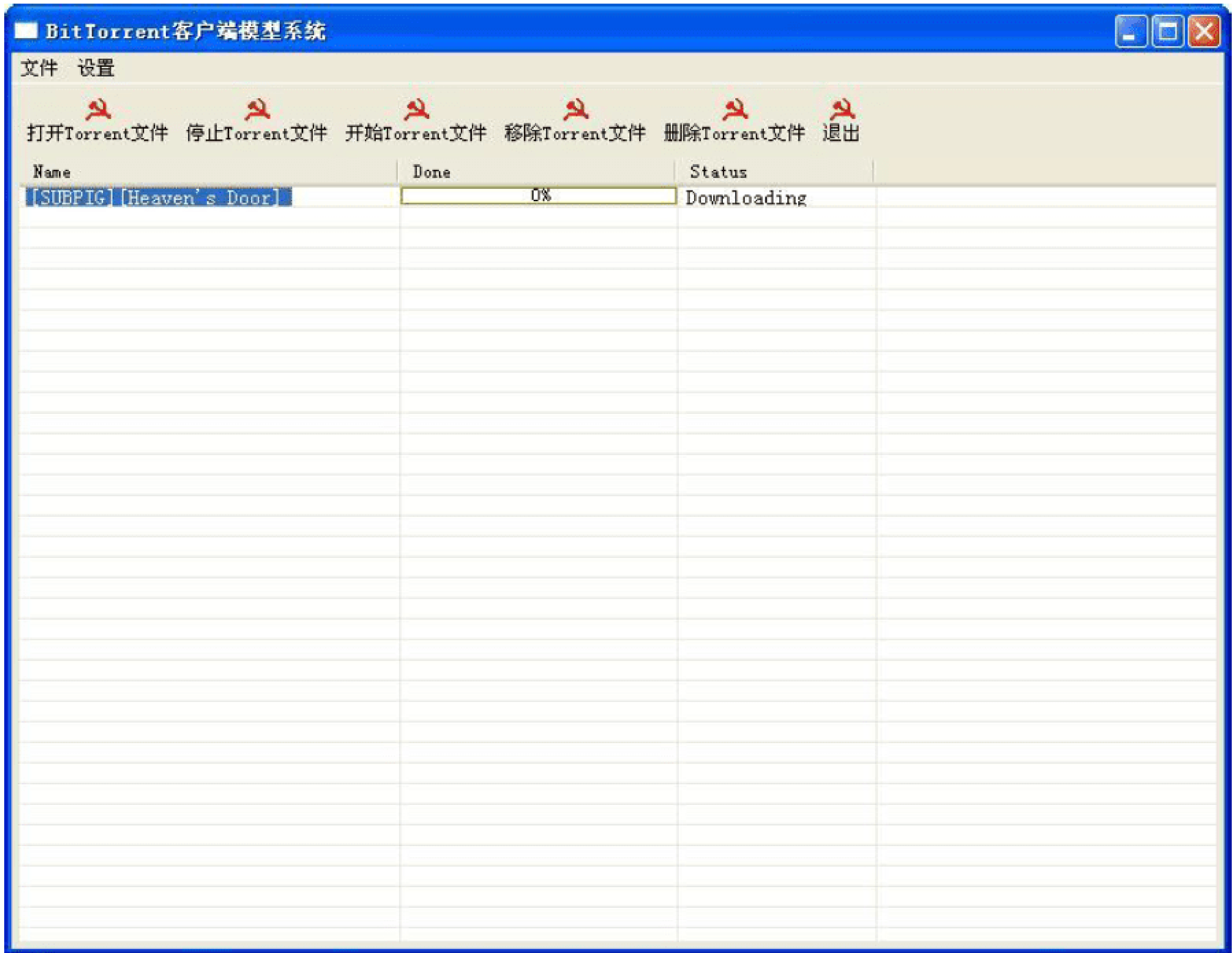


图 13.9　bitclient_model 工程运行的界面截图

13.7　本 章 小 结

本章主要分析了 BT 系统模型，讲解了 BT 客户端系统的一般开发方法，读者要重点理解整个 BT 系统搭建方法、构成系统的基本元素、各实体之间的相互关系等，还要了解整个系统的工作流程和原理。

在 BT 客户端系统的开发上，要了解一个普通的 BT 客户端系统的基本功能、所实现的协议及要完成的核心工作流程，同时，要重点掌握.torrent 文件的解析方法、Peer 与 Tracker 的交互方法以及 Peer 之间的通信过程。

BT 是 P2P 技术的重要体现，不管在当前网络生活还是在商业应用中都有突出的表现，掌握 BT 整个系统模型的搭建方法和 BT 客户端的开发技术，对自己的学习和未来发展都有重要的现实意义。

第 14 章　P2P 的解决之道——JXTA 技术简介

P2P 应用已经引起全世界数百万计算机用户的关注，从 Napster 到 Instant Messenger，P2P 应用已跻身当今互联网上最广泛使用的应用之列。

P2P 技术在发展的过程中，不可避免地存在许多障碍，其中主要的障碍就是这样的应用趋向于孤立和无标准的开发，同时 P2P 应用却缺乏一个共同的底层基础。而 JXTA 技术致力于提供一套所有的 P2P 应用程序都能使用的标准来清除这些障碍，JXTA 的协议是一种与网络和编程语言无关的协议，并允许应用程序的开发不依赖于集中式的系统服务器，标准化的 JXTA 协议就与同标准化的网络协议（如 TCP 和 HTTP）一样，真正实现 P2P 计算的目标。

在本书的最后一章，将带领读者一起学习一下 JXTA 技术的相关知识，重点学习 P2P 的 JXTA 解决之道。本章知识的重点如下。

- JXTA 概述：了解现有 P2P 系统的缺陷，认识 JXTA 及 JXTA 的历史和背景。
- JXTA 基础知识：了解 JXTA 的设计目标和层次结构，对 JXTA 的基本术语、与 Java 的绑定及与 XML 的关系要有一定的理解。
- JXTA 的协议：了解 JXTA 的两个核心协议及 4 个标准服务协议，并理解这些协议它们各自的作用是什么。
- JXTA 的部署与应用：掌握 JXTA 的部署方法，能使用 JXTA 的 API 开发简单的 JXTA 应用。
- JXTA 的重要概念：重点理解 JXTA 中关于 ID、通告、消息的概念，掌握它们各自的应用开发方法。

14.1　JXTA 概述

JXTA 是一个用来解决 P2P 计算的开放的网络计算平台。JXTA 的 P2P 平台使开发者能在其上建立 P2P 的应用，本节就简要介绍一下 JXTA 的基础知识。

14.1.1　现有 P2P 系统的缺陷和 JXTA 的出现

现有的 P2P 系统有一些缺陷，大多数 P2P 系统用来实现一个单一类型的网络服务（Napster 用来音乐文件交换、Gnutella 用来普通文件交换），由于不同的网络服务特性及缺少一个共同的底层基础，每一个供应商都使用不兼容的技术使它的用户同其他的 P2P 通信相隔离。一个很简单的例子，一个处理即时通信的应用程序对如何在其他 P2P 领域（P2P 文件交换等）进行操作是完全不可知的。另外，虽然在这些应用程序中许多通信是 P2P 形

式，但它们仍趋向于依赖大型的集中式服务器来提供服务的关键要素。

作为 P2P 的解决之道，不同的协议，不同的体系，不同的实现是当前 P2P 解决方案的精确描述。现今，开发者使用各种各样的方法论和途径来创建 P2P 应用。相对于客户/服务器模型丰富的标准，P2P 领域的标准可以说是很少，为此，Sun 开发了 JXTA，JXTA 的出现，为 P2P 领域的应用开发提供了一个简单的普遍底层平台，是 P2P 的重要解决之道。

14.1.2　JXTA 是什么

JXTA 是一个针对点对点（P2P）计算的合作研究项目——JXTA 项目（或简称 JXTA），它提供一套简单、精简和便利的技术，可以在任何平台、任何地点和任何时间支持 P2P 计算。

注意：JXTA（发音作 juxta）是 Sun 微系统对等网络（P2P）的标准，这是一个努力的方向，以它来促进和探究分布式计算的新方法。JXTA 这个名字既用来指代这个标准，也用来指代研究出来的技术，这种技术处于传输平台和 P2P 通信协议的环绕之中。其开发组织被取名为单词 juxtapose（并置）的简易形式。在 2001 年 2 月由 O'Reilly Network 组织的 P2P 会议上，Bill Joy，这位 Sun 的首席科学家解释了选择这个名字的原因：并置就是要让东西彼此相邻地放置，而“这正是对等网络的全部意义”。

JXTA 提供了一个基本的 P2P 架构，其他的 P2P 应用可以建立在上面。这一基本架构包含有一套实现 P2P 计算的公共协议，每一种协议都易于实现并集成到现有的 P2P 服务和应用系统中。这样，不同的 P2P 系统之间可以方便地互相通信，协同工作，向对方提供服务。

JXTA 是与语言无关的，它不是 API，可以独立于编程语言，如 C 或 Java，也是与平台无关的，独立于系统平台，如 Windows 和 UNIX，虽然有标准的语言绑定可以用于 JXTA 协议，但是绑定不是强制的。

JXTA 是与网络无关的，JXTA 协议能用于 TCP/IP、HTTP、蓝牙、家庭网络等进行传送，也能在任何数字设备上实现，包括传感器、消费电子产品、PDA 设备、网络路由器、桌面电脑、服务器和存储设备。位于不同网络的对等体可以容易地采用标准 JXTA 协议进行通信。

总地来说，JXTA 是一组独立于语言和网络的协议，它使得在不断变化的协议作计算设备上开发应用程序和服务成为可能。

14.1.3　JXTA 的背景与历史

以上讲了 JXTA 的基本知识，对于第一次接触 JXTA 技术的读者而言，可能还是不太明白这一技术到底是干什么用的，下面就简要介绍一下 JXTA 的背景知识，希望借助这些背景能加深对此技术的理解。

JXTA 最早起源于 2000 年的夏天，目标是要解决几个技术与商业上的难题。2000 年，是 P2P 突飞猛进的高潮年，但高潮背后却是许多小公司用自己的封闭系统试图在 Internet

上圈一块地。Sun 认为，只有互通才能真正发挥出 P2P 的优势，就好像 IM（Instant Messaging），能互连的人越多，越有价值。所以 Sun 决定出面发布一个平台，使所有 P2P 系统都能连接起来，于是，在 2001 年 4 月，SUN 发布了第一个原型实现的平台，它是基于 JDK1.1.4 的，而这一平台，就是现在所说的 JXTA。

JXTA 的创始人，是 Sun 首席科学家 BillJoy，他所创建的 JXTA 技术是他二十多年酝酿的结晶。JXTA 技术提供了基础性的机制解决当前分布计算应用中面临的问题，实现新一代统一、安全、互操作以及异构的应用。JXTA 通过 Java 技术和 XML 数据表达的结合，提供了强大的功能使得垂直应用得以交互，并且可以克服目前 P2P 软件中的限制。同时，通过小型、简单、便于开发的构造模块，JXTA 将使开发者从建立各自框架的复杂工作得以解放，可以潜心关注于建设各类新颖、创造性的、分布式计算的应用。

为了鼓励和支持该技术的发展，JXTA 项目采用了开放源码的方式，因此吸引了大量业界人士参与到 JXTA 技术的研究与应用中。JXTA Community（www.jxta.org）就是人气很旺的一个 Java 技术研究开发的网站，此网站上有关于 JXTA 较全面而又详细的技术介绍，要学好这门技术，这个网站是不可或缺的资源。

可以说，JXTA 是一个全新网络编程和计算的平台，有效地解决了现代分布计算尤其是点对点（P2P）计算中出现的问题。当前，有很多基于 JXTA 平台的 P2P 应用系统，它们提供了使用户更便捷地访问连接在互联网上的个人电脑资源的新框架，从而进一步拓展互联网的空间。

14.2　JXTA 的基础知识

JXTA 作为门技术，它有自己的一套知识体系，知识点众多、内容丰富，本章不深入展开来讲，只是对 JXTA 作入门式的讲解。本节就介绍一下 JXTA 的几个知识点，包括设计目标、层次结构及基本的术语等。

14.2.1　JXTA 设计目标

JXTA 是为了构建 P2P 网络而制订的一组协议，是处理构建 P2P 网络所碰到的问题的解决方法，JXTA 标准协议规范介绍如下。

JXTA 由 6 个协议组成，这些协议是专为特定的、分布式的、对等的网络计算而设计的。使用这些协议，Peer 可以互相合作来建立自我组织、自我管理的对等组，而不必关心它们在网络中所处的位置（在网络边缘或者防火墙的后面），并且也不需要集中的管理机构。

因此 JXTA 的核心是 6 个协议，其次，JXTA 是 P2P 应用程序开发的运行平台。目前 JXTA 首先推出了基于 Java 的参考实现，提供了支持 6 个协议的 Java API，JXTA 还将推出包括 C 语言在内的其他编程语言的 API。JXTA 在设计时有如下几个目标：

- 与操作系统无关；
- 与编程语言无关；
- 为 P2P 应用提供服务和基础。

从本质上讲，JXTA 的目标是希望任何设备（从台式机到 PDA、汽车、洗衣机等设备）

都可以支持 P2P 编程。这里有几个概念上的目标，它们包括：

- 使用组（group）来组织对等体（Peer）并且在组内提供服务和应用的环境。
- 组可以使用认证和验证方式来控制组内的访问权限。
- 通过网络来发布关于对等体和网络资源的信息。
- 通过系统来发布各种请求。
- 提供一个基础平台，供对等体之间做路由和通信。在防火墙或者其他 NAT 后面的 Peer 之间的通信也是这个目标中关键的一部分。
- 提供一种机制允许对等体之间可以彼此监视状态和资源。

除此之外还有一些其他目标，例如加密、支持不同的通信协议、易用性、稳定性和性能等，所有这些目标在设计 JXTA 协议和最初的 Java API 时，都被考虑到。另外，开发人员和 Sun 公司的管理者还考虑了以下目标：

- 系统应该允许任何设备直接加入到 JXTA 网络中去。
- 系统应该允许 ISP 对网络上的 Peer 进行集中管理。
- 系统应该支持数字产品版权的管理，例如购买的软件、音乐 CD、电影等。
- 封装和抽象一些特定的核心功能，以便产生出商业方面的应用。

从上面列出的目标可以看出两点，首先要让企业觉得使用 JXTA 可以使自己对系统进行控制，原因在于大部分 P2P 系统没有集中式的管理，所以在应用中不受企业的欢迎；其次，对于硬件或者软件提供商来说，JXTA 系统需要能够创造出利润。

根据以上这些目标，JXTA 被设计成企业可以接受的、易维护的、健壮的，并且能够满足任何 P2P 应用的技术。

14.2.2　JXTA 的层次结构

JXTA 被设计成同标准化的网络协议如 TCP 和 HTTP 一样的协议，它也是分层次结构的，JXTA 平台主要被分为 3 层，其分层结构如图 14.1 所示。

第 1 层：JXTA 核心层，它包含了服务所需要的核心功能；

第 2 层：服务层，它提供了访问 JXTA 协议的接口；

第 3 层：应用层，它使用服务来访问 JXTA 网络和 JXTA 提供的功能。这样的设计和一个标准的操作系统比较相似，标准的操作系统包括核心操作系统、服务和应用程序。

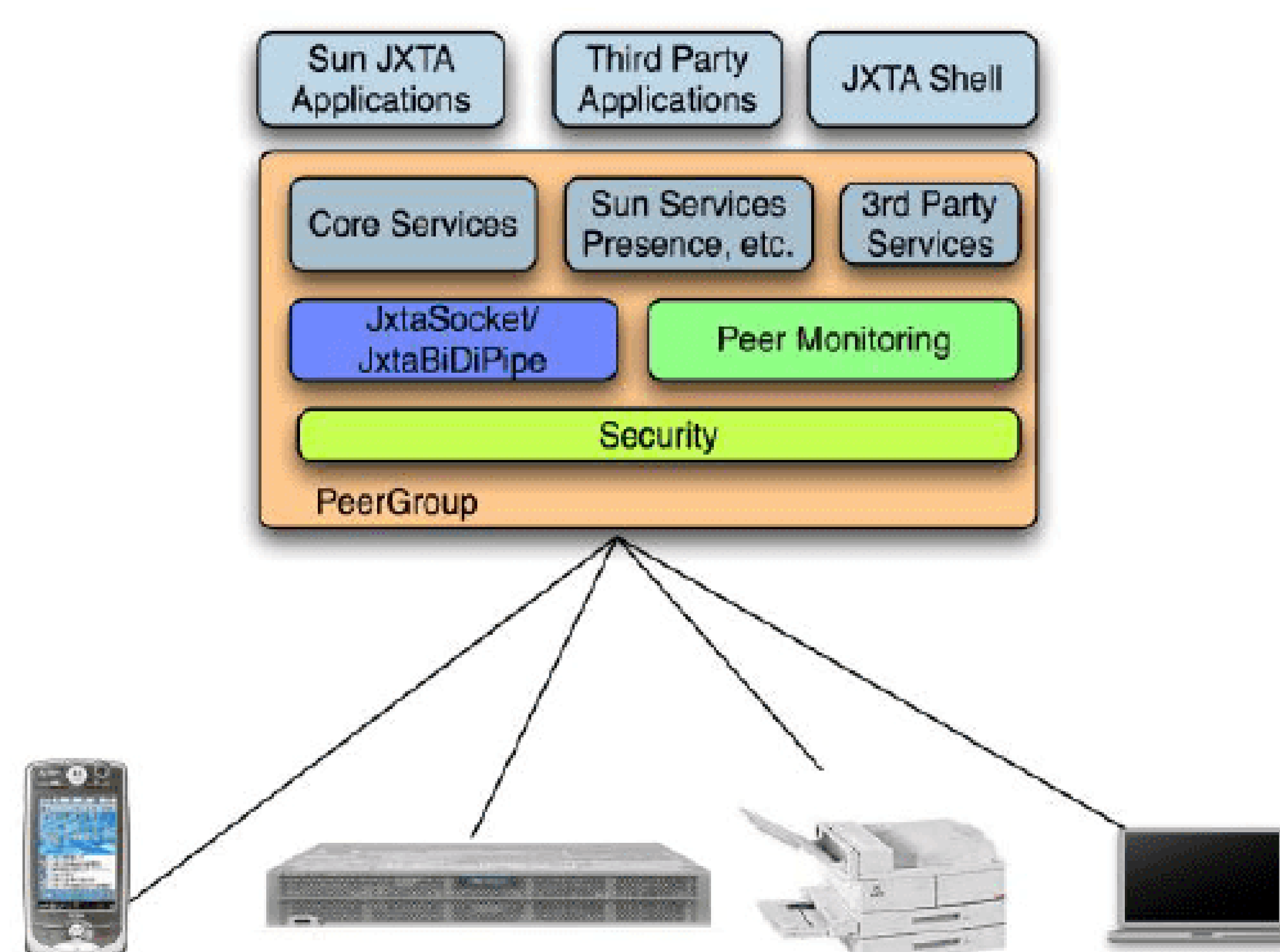


图 14.1　JXTA 中的分层结构

图 14.1 所示的 JXTA 的各个分层中，总结一下，各层有如下基本功能。

- 核心层（JXTA Core）：这一层封装了最根本的东西，包括 Peer、对等组、Peer 发现、Peer 通信、Peer 监视和相关的安全原语。
- 服务层（JXTA Services）：这一层包括对于 P2P 网络不是必需的、但很通用的功能，如查找、共享、索引、代码缓存和内容缓存的机制，以及认证、PKI 服务等。
- 应用层（JXTA Application）：这一层包括了应用 JXTA 服务开发出来的完整的 P2P 应用程序。例如 P2P 的即时通信系统、文件分发和资源共享系统、媒体内容的管理和传输，P2P 的 E-mail 系统，分布式的拍卖系统和其他的基于 P2P 的应用程序等。

JXTA 中虽然是以这种方式来分层的，但各个层之间并没有严格界线，在实际应用中，一个用户的应用可能被视为针对其他用户的服务。一个完整的系统都是基于模块化的思想来设计的，它允许开发者自己去挑选任何满足他们需要的一组服务和应用。

14.2.3　JXTA 的基本术语

JXTA 的技术手册中，列出了很多术语，理解这些术语是学习 JXTA 的基础，下面就列举出这些术语，并简要地说明它们所代表的意思。

1．Peers（对等体、结点）

网络上实现一种或者多种 JXTA 协议的任何实体，都叫做 Peer 结点，一个结点可以是任何东西，例如大型机，小至一部移动电话，甚至是一个传感器。结点的存在是独立的，并且可以与其他结点异步通信。

2．Peer groups（结点组）

有着共同目的的结点可以集合起来形成的组就叫做结点组，结点组也叫对等组，它是一种组织 Peer 并且发布结点组内的特定服务的方式。结点组可以跨越多个物理网络域，可以被创建、加入和退出，在一个组里还可以更新一个组成员的关系。

3．Messages（消息）

在 JXTA 的网络中，所有的通信是通过发送和接收信息来实现的。这些信息称为 JXTA messages。在 JXTA 中处理 Message 有两种方式，一种是使用 XML 格式，数据都遵行 XML 标准被包装到消息里；另外一种是使用二进制格式。不管用哪种方式，它们都要符合标准的格式，这是交互的前提。

4．Pipes（管道）

Pipes 在 JXTA 的环境里建立起来的一个虚拟的传输数据的通道，JXTA 中的 Pipe 不是内存中的数据结构，也不是硬盘上的文件，而仅仅是封装在 XML 广告文档中的以 ID 为标志的标签或者名字。因此，JXTA 中的 Pipe 是抽象的概念，是一个虚拟的通信管道，结点通过它们来发送和接收 JXTA 的信息。

JXTA 中的 Pipe 具有以下特点。

（1）Pipe 可以绑定到任何端点（Endpoing）：根据通信的需要，可以任意次地绑定。

（2）Pipe 是单向的、不可靠的、异步的：不可靠性是指 JXTA 不保证 Pipe 传送的消息一定可达目的主机，也就是说传输过程中，消息有可能丢失。异步性是指同一个 Pipe 发送的多个消息有可能不是依次抵达目的主机的。

Pipe 是 JXTA 最基本、最重要的特性，它提供了一种很好的方案，使得 Peer 在大多数网络情况下都可以通信，而不用只管防火墙或者其他的障碍。即使你不知道另外一个 Peer 的位置及它所使用的协议等信息，通过管道仍然可以与之通信。

5．Services（服务）

JXTA 平台提供了一种基础设施通过结点发现协议来发布和发现网络服务。像任何一种 JXTA 资源一样，网络服务用服务通告来表示，服务通告是一个 XML 文档，其中包括了唯一识别某一服务的所有信息和调用该服务所必需的全部信息。对等组成员可以发布和发现服务通告，就像操作任何其他通告一样。

在 JXTA 平台中有两种基本的网络服务，分别为：

- 对等体（Peers）服务。
- 对等组（Peer groups）服务。

对等体服务只能由发布该服务的对等体来实例化。如果多个对等体都想提供该服务，那么在对等组内部可能会运行着该服务的多个实例；每个对等体都将分别通告该服务。通告对等体服务是通过发布遍及整个对等组的服务通告来完成的。

对等组服务包含着服务实例的集合，这些服务实例运行在对等组的多个成员上并潜在的彼此协作。当某个对等体发生故障时，因为还可以从另一个对等体成员那里获得服务，所以集体的对等组服务不会受到影响。对等体服务的通告是对等组通告的一部分。

6．Codats

Codat（Code/Data），在 JXTA 中，它意味着可以是代码或者数据的内容。如果有需要，Codats 可以被发布和复制。

7．Advertisements（广告、通告）

一个广告就是一个 XML 文件，它用来发布和揭露可以获得的任何 JXTA 资源。例如一个 peer，一个 peer 组，一个管道或者 codats，广告都遵守编码，大多数的 JXTA 广告的编码是使用 UTF-8，它是对 Unicode 的一种 ASCII 编码方式。

8．Identifiers（标识符）

在 JXTA 环境中，Identifiers 扮演着重要的角色。Identifiers 指定资源，而不是物理的网络地址。JXTA 的 Identifier 被定义为一个 URN（Uniform Resource Name，统一资源名）。一个 URN 就是一个 URI（Uniform Resource Identifier，统一资源标识符），它必须是保持全局唯一，即使是该资源不存在了，它仍然要保持不变。

9．World peer group

任何一个 JXTA peer，在默认的情况下都是 World peer group 的一个成员。每一个 JXTA

peer 都知道 World peer group，并且可以加入该 peer 组，即使它在网络上找不到任何其他的 peers。即使断开的 peer 也是其中的成员。

10．Net peer group

在一个本地网络中，网络管理员通常都可以设置一个 peer 组，该组可以让网络上的任何 peer 加入，这就是 Net peer group。它类似一个 DHCH（dynamic host configuration protocol，动态主机配置协议）服务。该 Net peer group 可为 peer 提供一个全局的连接，它的限制由管理者定义。

11．Rendezvous peers（集合点、汇聚点）

一个集合点首先是一个 Peer，而且是一个能够处理来自其他 Peer 请的 Peer。集合点是一个特别的结点，它通过缓冲结点的广播，可以存储其他结点的信息。因此，一个集合点可以帮助结点发现网络上的其他结点。集合点还可以作为搜索的传递者，将搜索的请求导向到其他的集合点。

使用集合点的一个主要目的就是为了方便在本地网络之外搜索广告。集合点通常拥有更多的资源，并且可以存储大量有关它周围的 Peer 信息。

12．Endpoints（端点）

在 JXTA 应用中，端点是最基本的通信方法。一个端点就是实现了特定通信协议的 Peer 的地址。一个 Peer 可以有多个端点，这样可以通过不同的协议来与其他的 Peer 通信。

端点描述了协议和连接的所需要的信息。因此端点可以描述 HTTP、TCP 以及其他可以支持的通信协议，这样，Peer 可以提供更有效率的方法。也就是说，如果两个 Peer 都在防火墙的后面，可以直接通过它 TCP 端点来通信；如果两个 Peer 要穿过防火墙去通信，则需要使用 HTTP 端点。

13．Routers（路由器）

在 JXTA 的网络上移动包的东西均被称为 JXTA 路由器。并不要求所有的 peers 都是路由器。不是路由器的 peers 必须找到一个路由器来路由它们的信息。

除了以上所说的这些术语外，JXTA 中还有很多其他的术语，读者可根据自己的学习需要参考相应的文档。

注意： 以上术语的解释将来自 JXTA 的英文参考文档，由于对专业术语的理解和翻译习惯的不同，在不同的文献上有不同的中文命名方法，读者如有疑问的地方，请自行参考 JXTA 技术文档。

14.2.4　JXTA Java 绑定

JXTA Java 绑定，它是用 Java 实现的一个 JXTA 参考。开发者可以在现有的实现上建立应用或者选择自己的语言和平台来实现这些协议。在参考中，为了简化，使用 HTTP 和 TCP/IP 来传送，不过也可以根据网络的拓扑，使用任何的传送协议来实现 JXTA 协议。

1．JXTA java绑定中类的组织

JXTA 的 Java 绑定包含有两个主要的类层次：

- The net.jxta.* classes；
- The net.jxta.impl.* classes。

第一个包包含所有的 JXTA 接口，它们是 JXTA 协议和核心建立块的 Java 接口。第二个包是这些接口的实现。这些接口和它们的实现必须是完全分开的。让我们来看一下这些包。

2．如何找到结点

结点是网络中的一个独立的、异步实体，与一个结点 ID 关联。你可以将运行代码的一个实例看成为一个结点。在 JXTA java 绑定中，通过一个启动类来启动一个结点运行。

注意：此启动类为 net.jxta.impl.peergroup.Boot，它内部带有一个 main()方法。读都可在此路径下查看 Boot 类的源代码。

一个结点的性能与它从属的组有关。不过，每个结点都有一些基本的性能，例如拥有一个 ID。这意味着存在至少一个 world 结点组，因为每个结点都必须是该结点组的一个成员，它也被称为平台结点组。world 结点组由 net.jxta.impl.peergroup.Platform 来表示，它是结点组类（net.jxta.peergroup.PeerGroup）的一个实现。

3．关于结点组

JXTA 中的结点组，有两个问题需要注意，一个是结点组嵌套，另一个是作为应用的结点组。

（1）结点组嵌套

下面说明一下，为什么要使用结点组嵌套，以及如何进行结点组嵌套。

举个例子，假设 P1 是结点组 PG1 的一个成员，该组提供一个基本的发现和搜索服务。在网络中有另一个结点组 PG2，它提供一个更高级的搜索服务。P1 必须加入 PG2 以利用其高级搜索服务，同时它也使用 PG1 提供的发现服务。要做到这一点，就可以使用结点组嵌套。

通过结点组的嵌套，一个结点组的服务可过载一个或者多个其他结点组的服务。这样我们就可以得到一个继承的关系：第一个结点组，在这里是 PG1，它作为一个父组，而第二个 PG2 则是一个子组。在这里子组的搜索服务重载父组的相应服务。因此，当 P1 加入 PG2，你可以知道结点组的嵌套是 WorldPeerGroup/PG1/PG2。要注意到 world 结点组一直在最顶层，所有的结点隐含都是该组的一个部分。

（2）作为应用（applications）的结点组

在绑定中有一个很重要的抽象，一个应用（net.jxta.platform.Application）就是一个结点组可以初始化、启动和停止的东西。要注意的方面是一个结点组（net.jxta.peergroup.PeerGroup）通常会启动另一个结点组（refer to the discussion on peer group nesting），因此应用也是这样。例外的是平台（或者称为 world）结点组，它并不由任何其他的结点组启动，它构成了结点组继承关系的基础部分。一个应用定义了 3 个方法，即 init()，startApp()，and

stopApp()。

在 Application 类中有以下几个方法：

- public void init(PeerGroup group, Advertisement adv);
- public int startApp(String[] args);
- public void stopApp()。

4．管道和终点

上面已经提到过，管道是虚拟的通信通道，它用来在 JXTA 环境中传送信息。管道由管道接口（net.jxta.pipe.Pipe）代表，它被认为是一个服务，因此由 Service 接口（net.jxta.service.Service）继承而来。管道由可以分成为输入管道（net.jxta.pipe.InputPipe）和输出管道（net.jxta.pipe.OutputPipe）。PipeService 类(net.jxta.impl.pipe.PipeService）中包含有在绑定中用到的管道实现。

管道是在终点上实现的。终点类（net.jxta.impl.endpoint.Endpoint）是传送终点的一个集合。终点可以使用 propagate()方法来广播信息，或者使用一个 endpoint messenger（终点使者）实现来发送信息到一个指定的终点（net.jxta.impl.endpoint.EndpointMessenger）。终点使用的实现是根据传送的协议来进行的。例如，net.jxta.impl.endpoint.http.HttpNonBlockingMessenger 类是一个实现 HTTP 协议的终点使者。

5．Advertisements（广播）

在绑定中，所有的广播都是由抽象的超类 Advertisement（net.jxta.document.Advertisement）扩展而来。根据广播表示的资源类型，它还可以进一步分为结点组广播（net.jxta.protocol.PeerGroupAdvertisement）、管道广播（net.jxta.protocol.PipeAdvertisement）等。AdvertisementFactory 类（net.jxta.document.AdvertisementFactory）创建广播。factory 可用来隐藏广播的真正实现。

6．服务

所有的服务都是实现 Service 接口的（net.jxta.service.Service）。一个服务就是一个应用，因此它扩展 Application 接口。因为服务对象不可以直接操作，所以一个服务接口通常访问一个 Service 对象。例如，Discovery 接口（net.jxta.discovery.Discovery）表示 Discovery 服务。DiscoveryService 类（net.jxta.impl.discovery.DiscoveryService）表示 Discovery 服务的实现。DiscoveryService 类不可以直接访问，而是通过一个 DiscoveryInterface 接口（net.jxta.impl.discovery.DiscoveryInterface）实现。

14.2.5　JXTA 与 XML

毫无疑问，要提供一个通用基本协议层，第一步就是要采用一种适合的表现方式，这种方式可以被当前的大部分平台承认。XML 无疑是一个理想的选择，它已经成为数据交换的一个默认标准。XML 提供通用的、语言和平台无关的数据表现。XML 也可以很容易地转换为其他的编码。因此，用 XML 格式定义了所有的 JXTA 协议。

XML 通过一个分布式系统提供了一个强大的表示数据和元数据的方法，并提供了通用

的表示方法，XML 主要有以下几个优点：

- XML 与语言无关；
- XML 是自描述的语言；
- XML 内容分类很明确；
- XML 确保正确的语法。

正是由于 XML 具有这些优点，对等体才能够安全地管理和使用通告信息，并确保与其他对等体之间的交互。

虽然 JXTA 的信息使用 XML 定义，不过 JXTA 并不依靠 XML 来编码。实际上，JXTA 实体并不需要一个 XML 解析器；它是一个可选的组件，可以将 XML 看成是 JXTA 使用的一种便利的数据表现形式，小的实体（例如移动电话）可以使用预编译的 XML 信息。

注意：在 JXTA 中使用 XML 作为发布通告的语言，除了由 XML 和 DTD 自己提供的语法检查之外，使用 XML 还可以对内容执行严格的类型检查，建议服务和协议的作者使用 XML 语言来指定通告或通告的子类型。

14.3　JXTA 的协议简介

JXTA 协议由 6 个为特定的、普及型的 P2P 网络计算而设计的协议构成。使用 JXTA 协议，对等体就可以自动加入动态对等组，这一过程独立于它们在网络中的物理位置（边缘、NAT、防火墙等），而且不需要集中管理的基础设施。本节就介绍一下 JXTA 的这 6 个协议。

14.3.1　JXTA 协议的分类

JXTA 的 6 个协议合作完成了对等体之间的路由、发现、组织、监控和通信。这 6 个协议总体上分为两类：

- 是核心协议；
- 标准服务协议。

1．关于核心协议

JXTA 被设计为只有少许必需的组件和行为一个小型系统。JXTA 的核心协议规范（JXTA Core Protocol Specification）中定义了所有实现都需要的最小限度的功能。核心规范定义了一个通用的通信层，该层使得对等体可以互相作用。任何与 JXTA 相兼容的具体实现只需要实现此核心规范即可。

JXTA 的核心规范主要描述了以下 3 个内容：

- ID 格式；
- 通告架构；
- 消息线格式。

JXTA 的 6 种协议中，核心协议有两个，分别为对等体端点协议（Peer Endpoing Protocol，

PEP）和对等体解析协议（Peer Resolver Protocol，PRP），关于这两个协议的详细内容在后文会有讲述。

总地来说，核心协议为对等体提供了在 JXTA 网络中进行连接、路由、发送和接收消息所需的最小限度的功能。

2．关于标准服务协议

JXTA 可能需要实现许多可选的协议，这些协议不是核心协议规范的一部分，而是标准服务协议规范（Standerd Services Protocol Specification）的一部分。为了使一个 JXTA 实现能够与其他实现进行相互操作，必须要实现某些标准服务协议。例如，JXTA 实现可能需要实现一个发现协议才可能去动态地发现资源，需要实现一个管道绑定协议才可支持管道的动态解析，或者需要实现一个对等体信息协议去收集监控数据。

因为标准服务协议是可选的协议，所以不要实现所有这些协议。例如，某一具体实现能预先配置为对等体可使用所有资源，那么就不需要发现协议了。某些具体实现可能不需监控，那么就不需要对等体信息协议了

标准服务规范详述了 4 个标准服务协议，分别如下。

- 对等体发现协议：Peer Discovery Protocol（PDP，结点发现协议）；
- 汇聚协议（Rendezvous Protoclol，RVP）；
- 对等体信息协议（Peer Information Protocol，PIP）；
- 管道绑定协议（Pipe Binding Protocol，PBP）。

下文会针对这些协议进行具体的讲解。

14.3.2　JXTA 核心协议规范

在上文已经说过，JXTA 的核心协议规范中主要描述了两个核心协议，一个是端点路由协议，另一个是对等体解析协议。

1．端点路由协议（EndPoint Routing Protocol，ERP）

端点路由协议，主要是被对等体路由器用来发送消息给另一个路由器来找出一个消息到达目的的路由。

JXTA 网络就其本性而言是一种特定的、多活跃结点的、自适应的网络。网络的连接可能是短暂的，而且消息的路由是不确定的。路由可能是单向的，而且变化很快，对等体可能频道的加入或离开。

位于防火墙后面的对等体可以直接向防火墙以外的对等体发送消息，但是防火墙以外的对等体不能直接与防火墙以内的对等体建立连接。

端点路由协议定义了一组由路由服务处理的请求/查询消息，该路由服务将消息发送到其目的地。当一个对等体需要向一个给定的对等全端点地址发送一条消息时，它先查看自己的本地缓存，看有没有到达该对等体的路由。如果没有找到任何路由，那么它就向可能的对等体路由发送一路由解析器查询消息，请求路由信息。

对等体可以使用它能发现的任意多的对等体路由器。对等体一般都有一个预告配置好的路由器列表。预先配置好的路由器是可选的。

ERP 为对等体提供了最后可采取的路由，通过用更复杂的路由服务代替核心路由服务，可以实现更智能的路由。高层的路由服务能够比核心服务更有效地管理和最优化路由。

注意：这里的 ERP 是端点路由协议的意思，通常我们所知的另一种 ERP 的意思是 Enterprise Resource Planning（企业资源计划）的简写，这种 ERP 是指建立在信息技术基础上，以系统化的管理思想，为企业决策层及员工提供决策运行手段的管理平台，要注意这两个不同的概念。

2．对等体解析协议（Peer Resolver Protocol，PRP）

在网络中，结点通常都会发送查询到其他的结点，以定位一些服务或者内容。这些查询的格式是通过 Peer Resolver Protocol 来标准化的。通过这个协议，结点就可以发送查询和接收响应。

对等体解析协议允许一般的查询分发到对等组中的一个或多个处理程序，并用响应来匹配这些查询。虽然每个查询都寻址到某个特定的处理程序的名称，该名称定义了该查询和其响应的特定语义，但是查询不与任何特定的对等体联系在一起。

在对等体解析协议中，给定的查询可以被对等组中任意数量的对等体接收，如果处理程序的名称在该对等体上注册了，那么就可以根据该处理程序的名称对给定的查询进行处理。

14.3.3　JXTA 的标准服务规范

JXTA 的标准服务规范中描述了 4 个标准服务协议，下面就分别介绍一下这 4 个协议。下文所讲的 4 个协议中，只对其中之一的结点发现协议进行详细的说明，其他的协议只做简要描述，关于 JXTA 协议的全部内容，请读者参考 JXTA 的协议规范。

注意：JXTA 协议规范可在 http://www.jxta.org 上下载，在本书的随书光盘中也提供了 JXTA 协议规范的英文原文。

1．Peer Discovery Protocol（PDP，结点发现协议）

对等机发现协议 PDP，主要用来发布自己的广告信息，和查找其他 Peer 的广告。如前文所述，Advertisement 是 Peer 之间各种信息交流的基本单元，发现其他 Peer 及其他 Peer 及其资源的问题就转换为发现描述各资源的 Advertisement 的问题，只要找到对应的 Advertisement，就相当于找到了该资源。

PDP 定义了发现其他 Peer 和资源的协议，该协议包括两个方面，一方面用于请求获得其他 Peer 的 Advertisement；另一方面用于响应其他 Peer 的这种请求。

（1）PDP 的消息格式

PDP 只由两种格式构成，它们是。

- 用于发现 Advertisement 的请求格式。
- 用于响应请求的响应格式。

这两种消息，即发现请求消息和发现响应消息，定义了两个 Peer 之间互相发现对方所

需的所有元素。下面分别介绍一下这两种消息。

虽然 discovery 消息定义了请求和对应的响应，但一定要注意，有时响应可能收不到，有很多原因会造成对应某请求的响应收不到，比如该请求未能生成任何结果或者响应超过一定数额被请求对方忽略了。

发现请求消息（Discovery Query Message）

一个 Peer 想要寻找某种 Advertisement 时，就会发出 Discovery Query Message（Discovery 请求消息），这个请求消息的格式如下：

```
<?xml version ="1.0" encoding ="UTF-8"?>
<jxta: DiscoveryQuery>
<PeerAdv>…</PeerAdv>
<Type>…</Type>
<Attr>…</Attr>
<Value>…</Value>
<Threshold>…</Threshold>
</jxta: DiscoveryQuery >
```

以上请求消息中标签的含义如下。

- PeerAdv：可选元素，表示 Peer 的 ID，如果设置为 null，则转向 Peer 所在 PeerGroup。
- Type：有 3 种类型，PEER（0）、GROUP（1）、ADV（2）。
- Attr 和 Value：都是可选元素，规定响应中的 Advertisement 需要满足的条件。前者指明 Advertisement 中的元素名，后者指明元素的值。只有满足这样的条件的 Advertisement 才能作为该请求的响应。
- Threshold：也是个可选元素，该元素包括一个数字，指明最多可以接受多少数目的 Advertisement。发出一个请求后，可能会有很多响应，指定了最多接受数目后，多余的会忽略掉。

注意：还有一种特殊情况，当 Type 设为 0 时，即请求获得 Peer Advertisement 时，将 threshold 也设为 0，发出这种请求意味着请求方想获得所有的 Peer Advertisement。这样收到请求的所有 Peer 都响应，提供自己的 Peer Advertisement。如果 Attr 和 Value 的值没有设定，其他 Peer 响应时回复的是 Advertisement 的随机集合，当然前提是符合 Type 指定的类型而且未超过 Threshold 元素规定的最大响应数目。

请求响应消息（Discovery Response Message）

为了回复 Discovery Query Message，Peer 创建一个 Discovery Response Message，其中包含符合请求中要求的 Advertisement，如 Attr/Value 要求或 Advertisement 类型的要求。

```
<?xml version ="1.0" encoding ="UTF-8"?>
<jxta: DiscoveryQuery>
<Type>…</Type>
<Count>…</Count >
<Attr>…</Attr>
<Value>…</Value>
<PeerAdv>…</PeerAdv>
<Response Expiration ="expiration time">
...
</Response>
</jxta: DiscoveryQuery >
```

以上响应消息中标签的含义如下。

- Type：与 Discovery Query Message 中的 Type 元素一样，只有 3 种类型，即 PEER（0）、GROUP（1）、ADV（2）。
- Count：一个包含 integer 值的可选元素，代表 Message 中响应元素的总数目。
- Attr/Value：一对可选元素，与 Discovery Query Message 中的值相同，指明此响应产生时对应的原始搜索条件。
- PeerAdv：一个可选元素，包含提供响应的 Peer 的 Peer Message。
- Response：一个可选元素，包含符合 Discovery Query Message 中搜索要求的 Advertisement。每个 Discovery Response Message 可以包含多个 Response 元素，每个元素只包含一个 Advertisement。而 Response 元素个数之和为 Count 的值。Response 元素中的 Expiration 属性指明 Advertisement 的有效时间长度，当然，这个时间是以毫秒为单位的。

（2）发现服务（Discovery Service）

服务的实例都与特定的 Peer Group 相关联。只有同一个 Peer Group 中的 Peer 才能通过服务来相互通信。默认情况下，所有的 Peer 都属于一个大的 Peer Group，称为 NetPeerGroup，因而所有的 Peer 及其 Advertisement 能相互发现。

Discovery Service 提供了如下功能：

- 获得远程的 Advertisement；
- 获得本地的 Advertisement；
- 将 Advertisement 发布到本地；
- 将 Advertisement 发布到远程；
- 释放本地 Advertisement。

DiscoveryService 接口为开发人员提供了简单的方法来发送 discovery 请求和处理 discovery 响应。使用 Discovery Service 提供的方法来发送 Discovery Query Message 甚至不需要开发人员创建和发布 DiscoveryQuery 对象。

（3）DiscoveryListener 接口

发现 discovery 请求后，接到响应时应用程序需要获得通知以便取得响应中的 Advertisement。在 Java 参考实现中，可以给 Discovery Service 注册一个 listener 对象，当 Discovery Response Messages 到达时可以通过 DiscoveryService 获得通知。关于 listener 在 Java 中的作用已介绍过，这里就不再详述。

每次 DiscoveryService 实例收到 Discovery Response Message，都会调用该 listener 的 discoveryEvent()方法，并传递事件本身，事件中包括了响应的细节。但 discoveryEvent()方法必须用到 DiscoveryEvent 类的参数。下面介绍一下 DiscoveryEvent 类。

（4）发现远程的 Advertisements

DiscoveryService 接口提供了简便的方法来给其他 Peer 发送 Discovery Query Message，不需要自己创建 DiscoveryQueryMsg 实例，而是用 getRemoteAdvertisements()方法。它有 5 个参数，第 1 个参数为 Peerid，如果为空，则会把请求发给本地的网络上的所有 Peer，并通过 Rendezvous Peer 对外广播。

（5）发现缓存中的 Advertisement

在 Java 参考实现中，响应一个 Discovery Query Message 的 Advertisement，会自动保存在本地的 Advertisement 缓存中，DiscoveryListener 在实现的时候不需要再写缓存的函数。

如果要用本地缓存发现 Advertisement，可以用 DiscoveryService 接口的 getLocalAdvertisements()方法。

与寻找远程的 Peer 不同，用 getLocalAdvertisements()方法可以马上得到结果，不需要用 DiscoveryListener 实现来处理在 DiscoveryResponseMsg 响应中返回的 Advertisement，马上返回一个与请求参数相匹配的 Advertisement 的 Enumeration。

在 net.jxta.impl.cm 包中，由 Cache Manager 类的 Cm 实现了本地缓存，处理发现的 Advertisement 的存储问题，存储在本地的目录结构中。

（6）清除缓存中的 Advertisement

有时，应用程序可能想清除所有的缓存，比如应用程序很久没有使用，估计所有 Advertisement 都已失效。通过下面的方法来清除缓存中的 Advertisement。

```
public void flushAdvertisement (String id,int type);
```

以上就是 JXTA 服务规范里的结点发现协议，除了这个协议外，还有另外 3 种协议，以下这 3 种协议只简要地说明一下，不再详细介绍，详细知识请参考 JXTA 的协议规范。

2．汇聚协议（Rendezvous　Protocol，RVP）

该协议为对等体提供了预订传播服务的机制。在对等组的内部，对等体可以成为汇聚对等体或者对汇聚对等体进行监听的对等体。RVP 允许对等体向服务的所有监听者发送消息。其他协议可使用 RVP 来实现高效地传播消息。

3．Peer Information Protocol（PIP，结点信息协议）

PIP 可以被用来 ping JXTA 环境中的一个结点。一个结点接收到一个 ping 信息，它的响应可以有多种：它可以给出一个简单的响应，里面只包含它的正常运行时间。它可以发送一个完整的响应，里面包含有它的广播；它也可以忽略该 ping。因此，可以有只能接收信息而不能发送响应的结点。

4．Pipe Binding Protocol（PBP，管道绑定协议）

在 JXTA 环境中，结点使用管道来访问服务。一个结点可以在运行时绑定到一个管道的一端来访问服务。结点可以创建一个新的管道，绑定到一个现有的管道，或者解除到某个管道的绑定。对于以上的操作，结点都要使用管道绑定协议。

14.4　JXTA 的部署与应用

上文中对 JXTA 的基础知识进行了简单的说明，学习 JXTA 的根本是在于应用。用 JXTA 进行 P2P 的开发，第一步就是要掌握 JXTA 的基本开发方法，如何部署、如何应用 JXTA 的 API，本节就重点讲一下 JXTA 的基本应用方法。

14.4.1　JXTA 的下载

JXTA 是开源的项目，网上提供 JXTA 的开发包、源码、例子等程序的免费下载，读者可到网站：http://download.java.net/jxta/上找到关于 JXTA 的所有资源的下载。

JXTA 当前最新的版本为 JXTA2.5，建议读者在下载的时候，将 jxse-doc-2.5.zip、jxse-lib-2.5.zip、jxse-src-2.5.zip 和 jxse-tutorials-src-2.5.zip，这 4 个软件包都下载下来，这 4 个包的内容分别如下。

- jxse-doc-2.5.zip：为 JXTA 的文档压缩包，这是应用 JXTA 的 API 重要的参考工具，是学习 JXTA 及进行 JXTA 有关开发的必要参考。
- jxse-lib-2.5.zip：是 JXTA 的类库，包括 4 个 Jar 文件，分别为 bcprov-jdk14.jar、javax.servlet.jar、jxta.jar 和 org.mortbay.jetty.jar。在进行 JXTA 的应用开发时，需要导入相应的 Jar 包。JXTA2.5 与 JXTA2.4 相比，类库做了整合，与原来的 8 个 Jar 包相比，减少了一半。
- jxse-src-2.5.zip：这是 JXTA 的源代码包，在进行 JXTA 开发时，有时候需要分析 JXTA 的相关源代码，那么这外包就显得很重要了。
- jxse-tutorials-src-2.5.zip：这是 JXTA 的一个开发指导实例的源代码，包括 JXTA 开发的指导文件档，在其网站上也能找到。

以上这几个软件包下载完成后，JXTA 也就下载完成了。下面就讲一下如何应用 JXTA 开发一个简单的 Hello World 程序。

注意：在 JXTA 的开发中，只用到它的类库包，将 JXTA 的 Jar 包导入到构建路径就能部署 JXTA 的开发环境。

14.4.2　JXTA 的安装与开发

JXTA 的安装过程其实就是导入 JXTA 开发包到具体的开发工程中的过程。安装 JXTA 有以下几个简单的步骤。

（1）将下载的 jxse-lib-2.5.zip 软件包解压，解压后此文件夹下将有 4 个 Jar 文件。在 Eclipse 中新建一个 Java 工程，名为 jxta_example，然后按照本书第 10 章所讲的将 Jar 包导入到工程的构建路径的方法，将 jxse-lib-2.5.zip 包内的 4 个 Jar 文件导入到工程的构建路径中。

注意：本例只写一个简单的 Hello World 程序，只需导入 jxta.jar 包就可以。

（2）在 jxta_example 工程中新建一个包，包名为 p2p.jxta.helloworld，然后再在此包内新建一个类，类名为 HelloWorld_JXTA，表示这是 JXTA 的第一个小程序。整个工程框架建好后，如图 14.2 所示。

（3）JXTA 的 Hello World 程序

下面来演示一下，如何利用 JXTA 写出第一个 JXTA 的应用程序。HelloWorld_JXTA 类，位于 jxta_example 工程中的 p2p.jxta. helloworld 包下，以下是 HelloWorld_JXTA 类中

主要方法的说明，详细完整的实现源代码请读者参考随书光盘。HelloWorld_JXTA 类的文件位置如下：

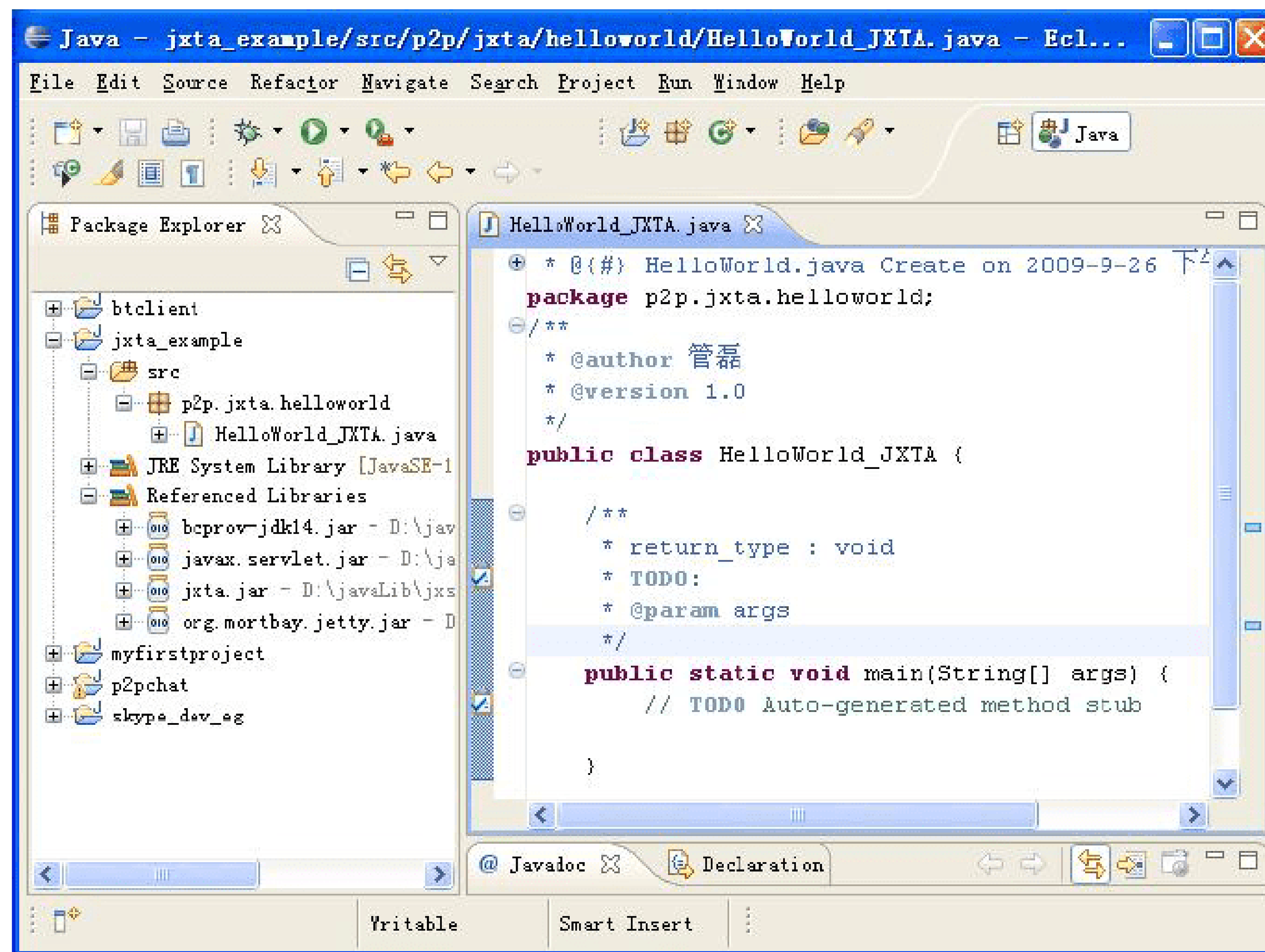


图 14.2　JXTA 开发工程框架图

【源代码：ch14\ch14_code\jxta_example\src\p2p\jxta\helloworld\HelloWorld_JXTA.java】

HelloWorld_JXTA.java 类的实现如下，在新建的 HelloWorld_JXTA 的类下，输入以下代码：

```
import java.io.OutputStream;
import net.jxta.peergroup.PeerGroup;
import net.jxta.peergroup.PeerGroupFactory;
import net.jxta.exception.PeerGroupException;
public class HelloJXTA {                          //定义测试类的名称为 HelloJXTA
  public static void main(String args[]) {  //类的主方法
    System.out.println("Starting JXTA....");//程序运行开始的时候打印一个字符串
    HelloJXTA myapp = new HelloJXTA();        //新建一个测试类的实例
    myapp.startJXTA();                        //调用 startJXTA()方法，启动 JXTA
    System.exit(0);                           //程序退出
  }
  public void startJXTA(){              //定义一个 startJXTA()方法，用于启动 JXTA
    PeerGroup pg = null;                      //初始化一个结点组对象
    try {
      pg = PeerGroupFactory.newNetPeerGroup();
                                              //通过结点组工厂新建一个结点组对象
    }catch (PeerGroupException e) {
       System.out.println("Fatal error:group creation failure");
                                              //处理异常情况
       e.printStackTrace();
       System.exit(1);                        //系统非正常退出
    }
    System.out.println("Hello JXTA!:)");      //打印一个字符串
    System.out.println("Group name = "+pg.getPeerGroupName());
                                              //打印结点组名称
```

```
    System.out.println("Group ID = "+pg.getPeerGroupID().toString());
                                        //打印组 ID
    System.out.println("Peer name = "+pg.getPeerName());
                                        //打印结点名称
    System.out.println("Peer ID = "+pg.getPeerID().toString());
                                        //打印结点 ID
  }
}
```

这样，JXTA 的第一个 Hello World 程序就写完了，这个程序中就包含了对 JXTA 最简单的引用。

14.4.3　JXTA 应用程序的运行与初始界面配置

运行以上源代码，第一次运行时，在 Eclipse 的控制台窗口会首先显示“Starting JXTA...”，然后出现 JXTA 的配置界面，如图 14.3 所示。在这个配置界面中，分别有 Basic、Advanced、Rendezvous/Relays 这 3 个设置页面，图 14.3 所示的是 Basic 页面，在这个界面中，需要填入以下几个选项。

- Peer Name：指结点的名称，随意填入一个名称即可。
- PassWord：指结点的密码，JXTA 规定，此处密码的设置应不少于 8 个字符。
- Verify Password：密码确认，确保两次输入的密码一致。

本例中设置用户为 jxta_test，图 14.3 所示的就是设置后的效果。接着单击 Advanced 按钮，进入 Advanced 的配置界面，如图 14.4 所示。

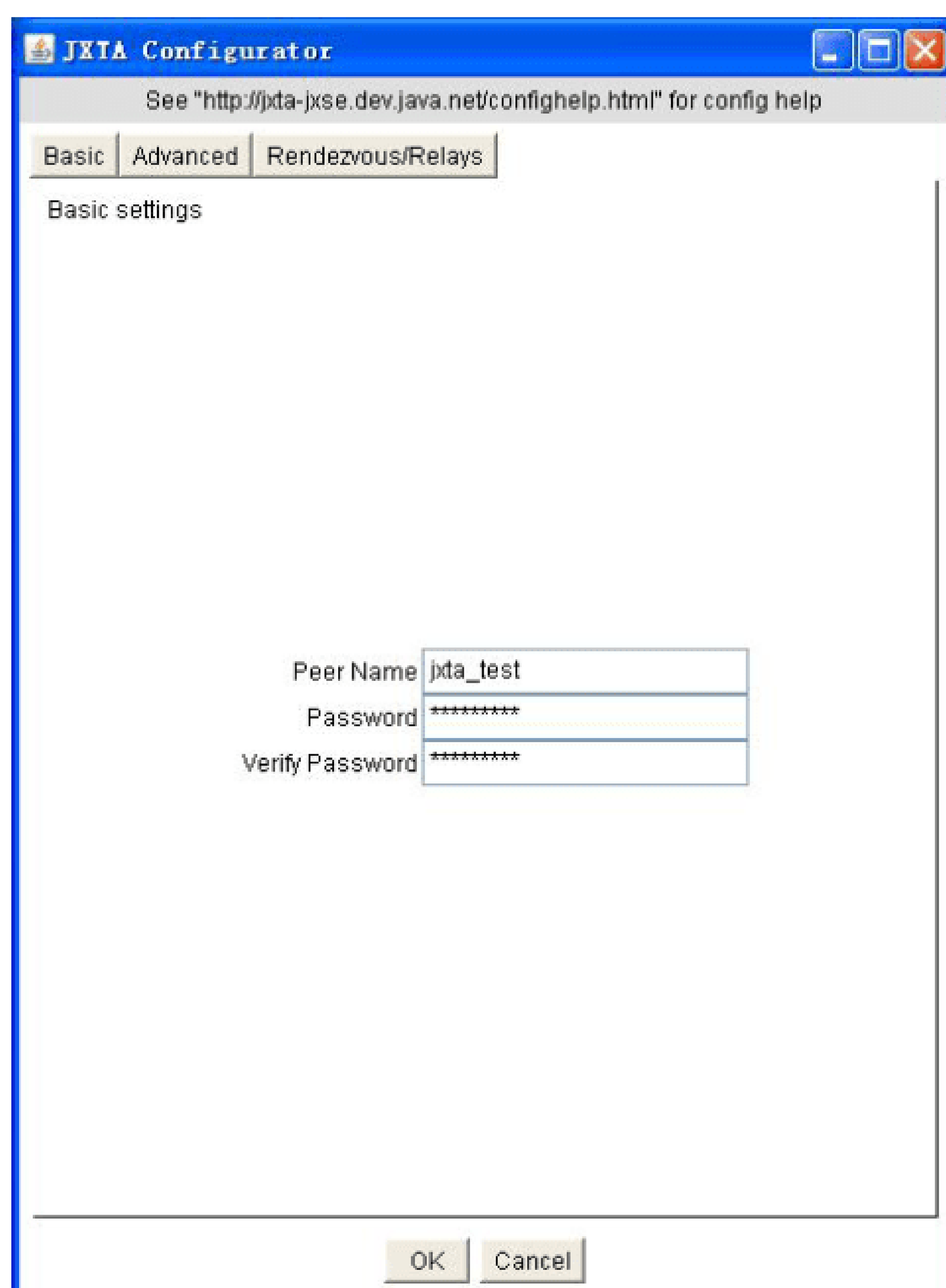


图 14.3　JXTA 初始运行时 Basic 配置界面

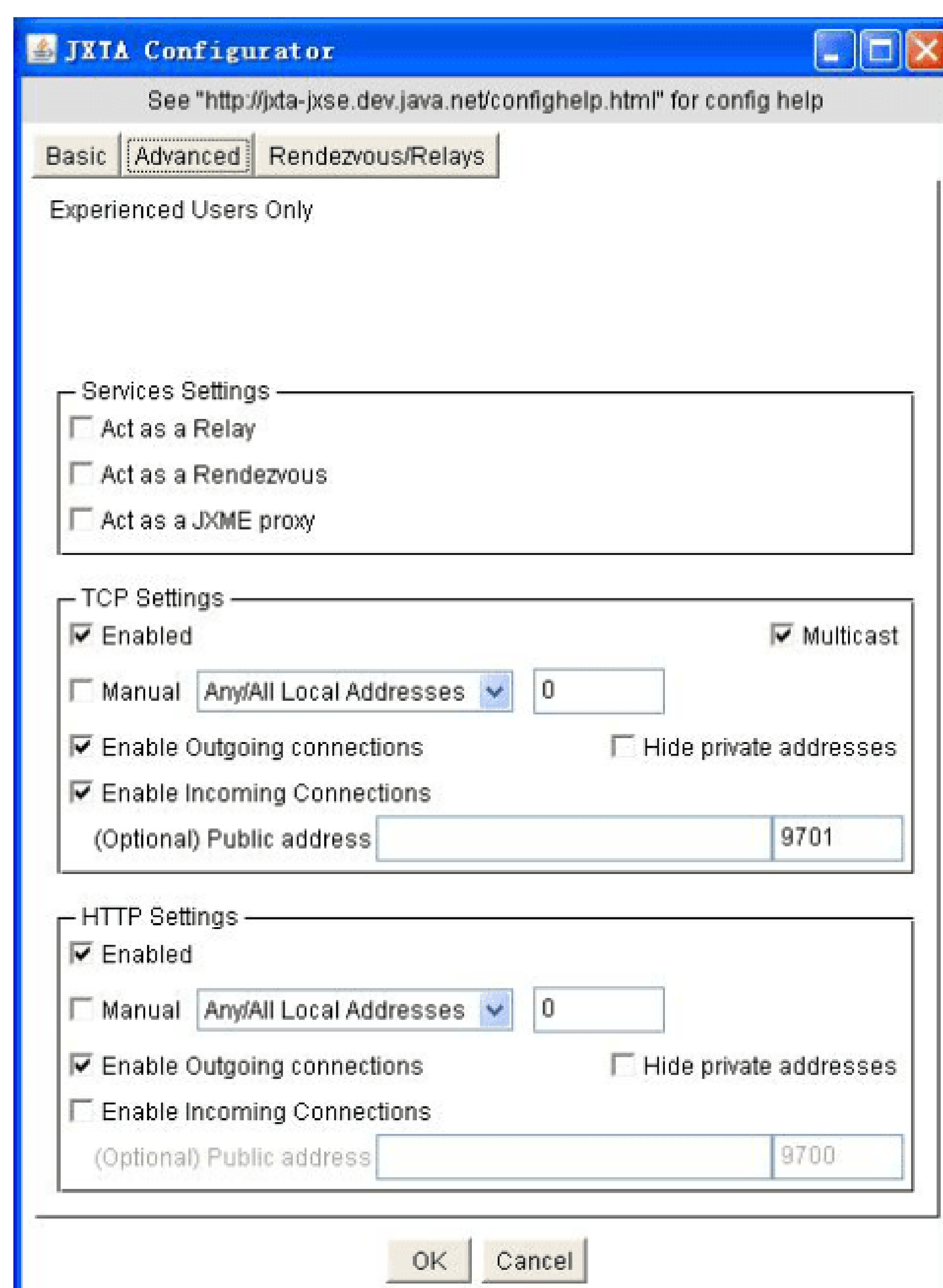


图 14.4　JXTA 初始运行时 Advanced 配置界面

在图 14.4 所示的 Advanced 界面中，可以设置以下几个选项：

- Services Settings：指的是服务设置，有 3 种服务选择分别为 relay Services、rendezvous services 和 proxy services。
- TCP Settings：在大多数情况下，TCP 的传输应该是允许的，即 Enabled 选项应该选中，同时，Outgoing connections 和 Icoming connection 也都设置成可用的。TCP 的设置在默认的情况下使用 9701 端口。
- HTTP Settings：如果处于防火墙或者 NAT（网络地址转换）的后面，就必须要使用 HTTP 的设置了。HTTP 的设置在默认的情况下使用 9700 端口。

图 14.4 就是在 Advanced 界面设置完成后的效果，再单击 Rendezvous/Relays 按钮，进入 Rendezvous/Relays 的配置界面，如图 14.5 所示。

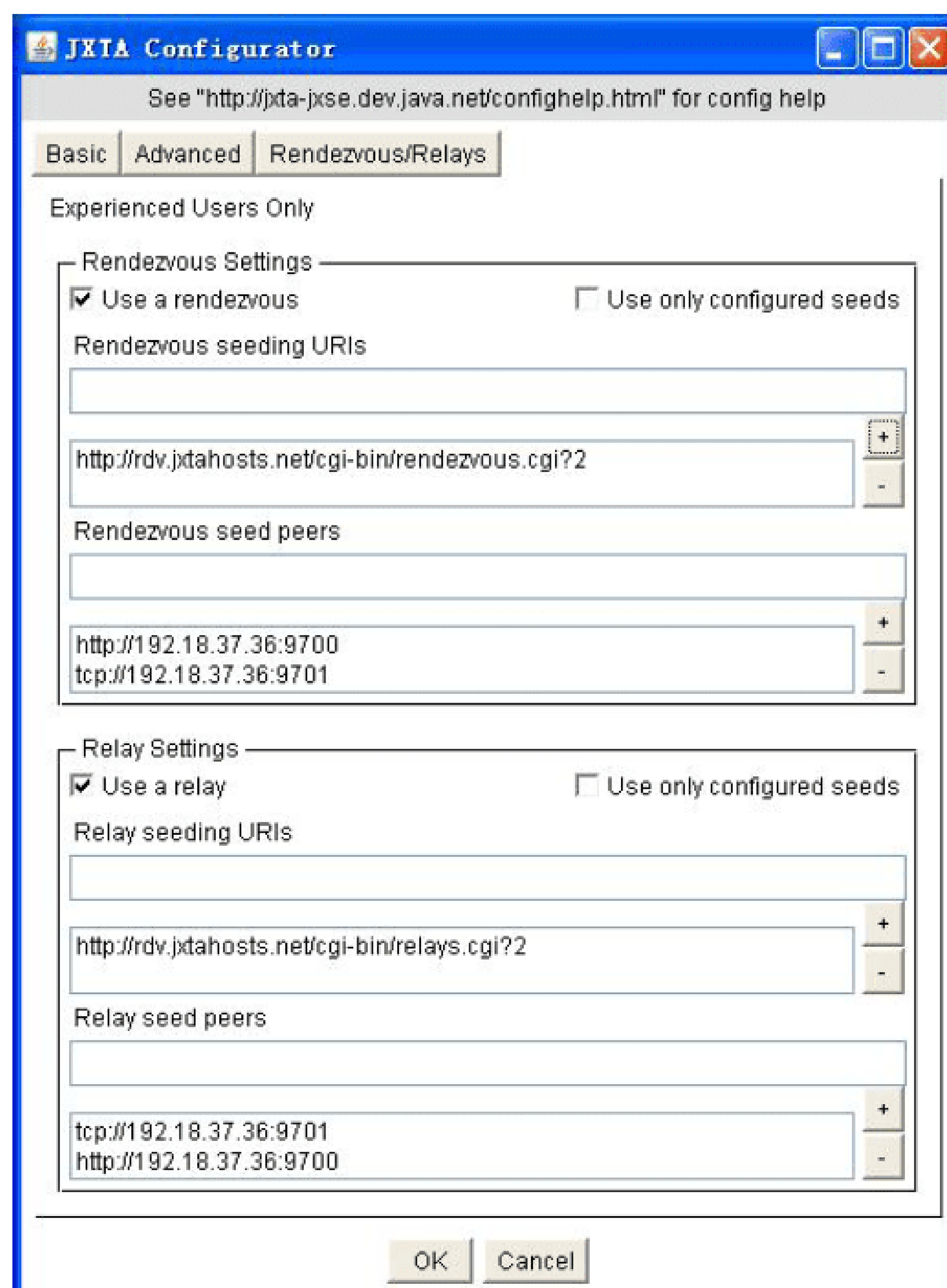


图 14.5　JXTA 初始运行时 Rendezvous/Relays 配置界面

在图 14.5 所示的界面中有两项需要设置。

- Rendezvous Settings：一般情况下都勾选 Use a Rendezvous，只有那些规定做种子结点的 Peer 才选择第二个选项，即 Use only configured seed。在 Rendezvous seed peers 选项区域中，设置一个汇聚的种子结点就行了，比如这里设置 Rendezvous seed peer 为 tcp://192.18.37.36:9701 http://192.18.37.36:9700
- Relays Settings：如果你的 Peer 不是可直接寻址的，那么就要勾选 Use a relay 这个选项，通常对一些边缘结点而言是必选的。在 Relay seed peers 的选项中，可将其设置为 tcp://192.18.37.36:9701，http://192.18.37.36:9700。

在 Rendezvous seeding URIs 和 Relay seeding URIs 中，如果是运行一个种子的 Web 服

务，就需要指定 URI，本例中将它们的 URI 分别设置为 http://rdv.jxtahosts.net/cgi-bin/rendezvous.cgi?2 和 http://rdv.jxtahosts.net/cgi-bin/relays.cgi?2。

注意：关于 JXTA 初始界面的配置，在 https://jxta-jxse.dev.java.net/confighelp.html 有详细的说明，读者可自己参考。

这样，JXTA 的初始配置界面就设置完成了，单击 OK 按钮后，HelloWorld_JXTA 程序接着执行，控制台中就会打印出如下信息：

```
Hello JXTA!:)
Group name = NetPeerGroup
Group ID = urn:jxta:jxta-NetGroup
Peer name = jxta_test
PeerID=urn:jxta:uuid-59616261646162614A78746150325033C5E44D4CF4D3435EB0
2B8466378B6C1103
Peer BaseClass = urn:jxta:uuid-DEADBEEFDEAFBABAFEEDBABE0000000905
```

这样，第一个 JXTA 的 Hello World 程序就编写和运行成功了。程序第一次成功运行以后，第二次运行时，就不会再弹出 JXTA 的初始配置界面了。程序运行完成后，会在程序同目录下会出现一个.jxta 的文件夹，其说明如下。

- PlatformConfig 文件：由 JXTA 的配置工具生成，是一个符合 XML 规范的文本文件。
- jxta.properties 文件：定义了一些 jxta 的属性。
- cm 目录：本地的缓冲目录，记录了发现的所有的 PeerGroup。在上例运行后发现 jxta-NetGroup 和 jxta-WorldGroup。
- pse 目录：存放 Peer 用于安全认证的证书信息。
- 如果将.jxta 的文件夹删除，则下次运行时和第一次运行时一样会出现 jxta 的配置界面。

14.4.4　JXTA 2.5 运行异常的解决方法

以上的 HelloWorld_JXTA 的程序，如果读者按上面的步骤一步步来操作的，那么程序在运行的时候可能出现一个异常。

运行这段代码的时候会出现如下错误：

```
警告:Failed to find class for urn:jxta:uuid-DEADBEEFDEAFBABAFEEDBABE0000
000C0206
java.lang.ClassNotFoundException:
No matching class for : urn:jxta:uuid-DEADBEEFDEAFBABAFEEDBABE0000000C0206
 at net.jxta.impl.loader.RefJXTALoader.findClass(RefJXTALoader.java:240)
 at net.jxta.impl.loader.RefJXTALoader.findModuleImplAdvertisement(RefJ-
 XTALoader.java:350)
 at net.jxta.impl.peergroup.StdPeerGroup.<clinit>(StdPeerGroup.java:143)
 at net.jxta.peergroup.WorldPeerGroupFactory.<init>(WorldPeerGroupFac-
 tory.java:178)
 at net.jxta.peergroup.NetPeerGroupFactory.<init>(NetPeerGroupFactory.
 java:205)
```

出现这个异常是由于 JXTA 2.5 的 Jar 包引起的，在 jxta.jar 包的内部有一个小 Bug，需要修正一下，具体的解决方法是：

在 jxta.jar 包里面的\META-INF\services\net.jxta.platform.Module 后面加上一句

```
urn:jxta:uuid-deadbeefdeafbabafeedbabe0000000C0206 net.jxta.impl.shell.
ShellApp Reference Implementation of Shell
```

在 Eclipse 中或者用普通的解压缩软件，将 JXTA 打开，找到 META-INF 文件夹下，services 文件夹里的 net.jxta.platform.Module 文件，此文件压缩状态下是不允许直接进行修改的。可以新建一个同名文件，将其内容全部复制后，在最后面加上文中的那句话，然后替换掉原来的文件即可。如果是将其完全解压后再修改，一定要记得还要将其打包成新的 Jar 文件后再导入到构建路径中方可使用。

14.5　JXTA 的重要概念及应用示例

JXTA 是一个针对 P2P 的解决之道，其主要功能体现在基于 P2P 的应用开发上，学习 JXTA 的重点是要掌握 JXTA 的一些重要概念及这些概念的理解和应用。本节就讲一下 JXTA 中的几个重要概念，并以实例代码的形式演示它们的具体应用。

注意： 本节所讲的源代码全部在 jxse-tutorials-src-2.5.zip 包中，此包是进行 JXTA 开发的一个引导示例包，建议读者认真地分析此包中的源码，本文只是选取几个频道 JXTA 重要概念的源码进行讲解。

14.5.1　JXTA 的 ID

JXTA 协议经常需要引用对等体、对等组、管道和其他 JXTA 资源。这些引用在协议中表现为 JXTA。JXTA ID 唯一性地确定了特定的对等组、对等体、管道、内容和服务实例。利用 JXTA ID，就可以对不同的 JXTA 实体进行清晰地引用。目前，有 6 种定义了 JXTA ID 类型的 JXTA 实体，分别为：

- 对等组；
- 对等体；
- 管道；
- 内容；
- 模块类；
- 模块规范。

JXTA 的 ID 通常表现为 URN 的形式，URN 是 URI 的一种形式，“用作不变的、位置无关的资源标识符”。和其他形式的 URI 一样，JXTA ID 是以文本的形式表示的。下面分别是 Peer ID 和 Pipe ID 的 URN 示例。

- JXTA 的 peer ID 示例

```
urn:jxta:uuid-59616261646162614A78746150325033F3BC6FF13C2414CBC0AB66366
6DA53903
```

- JXTA 的 pipe ID 示例

```
urn:jxta:uuid-59616261646162614E504720503250338E3E78229EA460DADC1A176B6
9B731504
```

在 JXTA ID 名称空间时，JXTA ID 是标准的 URN。JXTA ID URN 利用名称空间标识符 JXTA 来进行标识。JXTA ID URN 也都包含有 ID 格式关键字。ID 格式关键字表明了 JD 创建的方式，也可以使得 JXTA 绑定从 ID 中解析出其他信息。

在协议中使用 JXTA ID 时，它们被当作文本字符串 URI 进行处理。对 URI 有 3 种可行的操作，即比较、解析和分解。JXTA ID URI 可以被当作字符串来比较是否相同，也可解析为它们所引用的资源，最后，JXTA ID URI 能由 JXTA 绑定分析和解释。

无论格式或类型，所有 JXTA ID 都有以下属性。

- 明确性：必须是对资源的完整引用；
- 唯一性：只能引用单一资源；
- 规范性：对相同资源的引用应当编码为相同的 JXTA ID。这样就能够通过比较 ID 来判断它们是否引用同一资源，可理解性并不是所有的 ID 格式都能做到的。
- 不透明性：在 JXTA ID 的 URN 表述中，它们应当被看作是不透明的。协议消息中的 ID 的上下文足以确定其类型。如果 JXTA 绑定支持该 ID 格式，那么就可解释该 ID。

通过从根本上来讲，在 JXTA 协议里只有直接参考者才需要理解 JXTA ID 的内容。下面，就以程序实例的方式来演示一下，各种不同类型的 JXTA ID 都是如何构建的。

用来演示创建各种 JXTA ID 的类命名为 IDTutorial.java，此类位于 jxta_example 工程中的 p2p.jxta.tutorial 包下。以下是 IDTutorial 类中主要方法的说明，详细完整的实现源代码请读者参考随书光盘，IDTutorial 类的文件位置如下：

【示例源代码：**ch14\ch14_code\jxta_example\src\p2p\jxta\tutorial\IDTutorial.java**】

```
package p2p.jxta.tutorial;                    //声明程序所在的包路径
import net.jxta.id.IDFactory;                 //引入 JXTA 开发包中的 ID 工厂类
import net.jxta.peer.PeerID;                  //引入 JXTA 开发包中的 PeerID 类
import net.jxta.peergroup.PeerGroupID;        //引入 JXTA 开发包中的结点组 ID 类
import net.jxta.pipe.PipeID;                  //引入 JXTA 开发包中的管道 ID 类
import java.io.UnsupportedEncodingException;  //引入 I/O 异常处理类
import java.security.*;
import java.text.MessageFormat;
// IDTutorial 类，用来创建多种类型的 JXTA ID 的示例
public class IDTutorial {
     //定义一个静态、final 型的字符串常量值 IDTuorial
    private static final String SEED = "IDTuorial";
    /**
     * hash()方法，将传入的参数进行 SHA1 编码后返回
     */
    private static byte[] hash(final String expression) {
        //定义一个名为 result 的字节数组
        byte[] result;
          //定义此 MessageDigest 对象，实现消息摘要算法功能，信息摘要是安全的单向哈希
          函数，它接收任意大小的数据，并输出固定长度的哈希值
        MessageDigest digest;
```

```
    //判断输入参数的表达式是否为空，如为空则抛出异常
    if (expression == null) {
        throw new IllegalArgumentException("Invalid null expression");
    }
    try {
        //MessageDigest 类为应用程序提供信息摘要算法的功能，如 MD5 或 SHA 算
        法，这里用的是 SHA1 算法
        digest = MessageDigest.getInstance("SHA1");
     //捕获并处理异常信息
    } catch (NoSuchAlgorithmException failed) {
        failed.printStackTrace(System.err);
        RuntimeException failure = new IllegalStateException("不能得到
        SHA-1 消息摘要");
        failure.initCause(failed);
        throw failure;
    }
    try {
        //将输入字符串转换为 UTF-8 编码的字节数组形式
        byte[] expressionBytes = expression.getBytes("UTF-8");
        //调用 digest()方法，使用指定的 byte 数组对摘要进行最后更新，然后完成摘
        要计算
        result = digest.digest(expressionBytes);
     //捕获并处理异常信息
    } catch (UnsupportedEncodingException impossible) {
     RuntimeException failure = new IllegalStateException("不能将表达式
     编码成 UTF8 格式");
     failure.initCause(impossible);
     throw failure;
    }
    return result;
}
/**
 * createPipeID()方法，根据给定结点组 ID 和一个管道的名称，创建一个管道 ID，并
 返回
 */
public static PipeID createPipeID(PeerGroupID pgID, String pipeName) {
     //定义一个字符串 seed，值赋为 pipeName 和 SEED 连接
    String seed = pipeName + SEED;
     //调用 IDFactory 工厂类的 newPipeID()方法，创建一个管道 ID
    return IDFactory.newPipeID(pgID, hash(seed.toLowerCase()));
}
/**
 * createNewPipeID()方法，根据给定结点组 ID 创建一个新的管道 ID，并返回
 */
public static PipeID createNewPipeID(PeerGroupID pgID) {
    //调用 IDFactory 工厂类的 newPipeID()方法，根据传入的结点组 ID 创建一个管
    道 ID
    return IDFactory.newPipeID(pgID);
}
/**
 * createNewPeerID()方法，根据给定结点组 ID 创建一个新的结点 ID，并返回
 */
public static PeerID createNewPeerID(PeerGroupID pgID) {
    //调用 IDFactory 工厂类的 newPeerID()方法，根据传入的结点组 ID 创建一个节
    点 ID
    return IDFactory.newPeerID(pgID);
}
```

```
     /**
      * createPeerID()方法，根据给定结点组 ID 和一个结点的名称创建一个结点 ID，并返回
      */
     public static PeerID createPeerID(PeerGroupID pgID, String peerName) {
         //定义一个字符串的 seed，值赋为 peerName 和 SEED 连接
         String seed = peerName + SEED;
         //调用 IDFactory 工厂类的 newPeerID()方法,需要传入结点组 ID 和 seed 的哈希值,
         创建一个新的结点 ID
         return IDFactory.newPeerID(pgID, hash(seed.toLowerCase()));
     }
     /**
      * createNewPeerGroupID()方法，根据给定结点组 ID 创建一个新的结点组 ID，并返回
      */
     public static PeerGroupID createNewPeerGroupID(PeerGroupID pgID) {
       //调用 IDFactory 工厂类的 newPeerGroupID 方法，需要传入结点组 ID 来创建一个新
       的结点 ID
       return IDFactory.newPeerGroupID(pgID);
     }
    /**
      * createPeerGroupID()方法，根据给定组名称常量创建一个结点组 ID，并返回
      */
     public static PeerGroupID createPeerGroupID(final String groupName) {
         //定义一个字符串的 seed，值赋为 groupName 和 SEED 连接
        Stirng seed = groupName + SEED
        //调用 IDFactory 工厂类的 newPeerGroupID()方法，传入结点组 ID、默认的网络节
        点组 ID 和 seed 的哈希值，创建一个新的结点组 ID
         return IDFactory.newPeerGroupID(PeerGroupID.defaultNetPeerGroupID,
        hash(SEED + groupName.toLowerCase()));
}
```

以上所述的就是通过 IDTutorial 类来创建 JXTA 中各种 ID 的方法，这些 ID 包括管道 ID、结点 ID、结点组 ID 等。创建这些 ID 是进行 JXTA 开发所必需的步骤，读者可以写一个 main()方法来调用以上的方法，具体地验证 JXTA 中不同 ID 的具体创建过程。在随书源代码中，也有完整的程序代码，读者可运行程序自己演示一下。

14.5.2　JXTA 的通告

通告是 JXTA 规范用来描述资源的元数据文档。通告一般用于描述对等体、对等组、管道、内容、服务和许多其他类型的资源。XML 中引入了 JXTA 通告。许多 JXTA 规范都依赖于通告去提供密钥信息，或者用于两个对等体之间的通告传递。服务能够通过将已有的通告类型设为子类型或通过完全定义新的通告来定义新能通告类型。通告子类型能够提供附加的信息，以及丰富的元数据。

通告由一系列的层元素组成。这些元素按任意顺序出现在通告中，每个元素都包括它的数据或附加的元素，一个元素也可以有属性，这些属性是一些成对的名称/值字符串。其中有一个属性用于存储元数据，这样有助于描述元素中的数据。

JXTA 协议依赖于下面的通告。

- 对等体通告：此通告描述了该对等体的资源，主要用来保存该对等体的特定信息，如名称、对等体 ID、当前可用的端点地址和单个组服务想要发布的运行时属性等。
- 对等组通告：此通告描述了该对等组的特定资源，包括名称、组 ID、描述、规范

和服务参数等。

- 管道通告：此通告描述了一个管道，并由管道服务用来创建相关联的输入管道端点和输出管道端点。每个管道通告都包含着管道的一个可选的符号名和一个描述管道类型的信息。
- 模块类通告：此通告描述了模块的一个类，其目的是要描述一个特殊模块类 ID 所代表的东西。
- 模块规范通告：此通告描述了一个模块的说明，其目的是描述一个特殊的模块规范 ID 所代表的东西。
- 模块实现通告：此通告描述了模块规范的某个实现，给定说明的实现可以通过说明 ID 来搜索。通过环境的类型可以选择一个实现，也可以通过名称、描述或参数部分的内容来使用实现。
- 汇聚通告：此通告由一个充当给定对等组的汇聚对等体的对等体发布。汇聚通告由正在寻找汇聚对等体的对等体检索。

下面就通过实例来演示一下，如何利用 JXTA 的 API 来创建一个新的通告类型，并且通过一个通告工厂来注册一个新的通告。

本示例程序命名为 AdvertisementTutorial.java，此类位于 jxta_example 工程中的 p2p.jxta.tutorial 包下。以下是 AdvertisementTutorial 类中主要方法的说明，详细完整的实现源代码请读者参考随书光盘，AdvertisementTutorial 类的文件位置如下：

【源代码：ch14\ch14_code\jxta_example\src\p2p\jxta\tutorial\AdvertisementTutorial.java】

AdvertisementTutorial 类，一个简单的 JXTA 的通告开发示例程序，主要用来创建一个描述系统信息的通告，然后利用这个通告将本主机的一些属性信息表达出来。此类继承了 Advertisement 类，同时实现了 Comparable、Cloneable 和 Serializable 接口，以下是此类的声明、构造方法和相关变量的定义。

```
// AdvertisementTutorial 类的声明，既要继承也需要实现接口
public class AdvertisementTutorial extends Advertisement implements
Comparable, Cloneable, Serializable {
   //定义主机信息，通过 JXTA 的一个 Advertisement 将这些信息广告出去
   private String hwarch;                     //主机的硬件架构
   private String hwvendor;                   //提供商
   private ID id = ID.nullID;                 //广告的 ID
   private String ip;                         //主机的 IP 地址
   private String name;                       //类名
   private String osname;                     //系统名称
   private String osversion;                  //系统版本
   private String osarch;                     //系统架构
   private String inventory;                  //信息清单
   //以下是针对以上定义的主机信息，定义名称的标签，都是静态的 fianl 常量类型
   private final static Logger LOG =Logger.getLogger(AdvertisementTutorial.
   class.getName());
   private final static String OSNameTag = "OSName";
   private final static String OSVersionTag = "OSVer";
   private final static String OSarchTag = "osarch";
   private final static String hwarchTag = "hwarch";
   private final static String hwvendorTag = "hwvendor";
   private final static String idTag = "ID";
```

```
    private final static String ipTag = "ip";
    private final static String nameTag = "name";
    private final static String swTag = "sw";
    /*
     * 一个私有的、静态的 String 类型的数组常量，定义一个索引区，因为一个 JXTA
     * 通告必须被定义为可索引的，这里用一个字符串数组表示
     */
    private final static String[] fields = {idTag, nameTag, hwarchTag};
    /**
     * AdvertisementTutorial，无参数的构造方法
     */
    public AdvertisementTutorial() {          //没有参数，且方法体为空
    }
    /**
     * AdvertisementTutorial，以 Element 对象作为参数的构造方法
     */
    public AdvertisementTutorial(Element root) {
         //将传入的 Element 对象，强制转换为 TextElement
        TextElement doc = (TextElement) root;
        //对当前的广告类型进行判断，是否与 TextElement 的实例名称相同
        if (!getAdvertisementType().equals(doc.getName())) {
             不相同则抛出异常
            throw new IllegalArgumentException(
                  "不能够从含有" + doc.getName() "的文档中构造"+ getClass().
                  getName() +"类的实例")
        }
         //调用 initialize()方法，初始化 TextElement 对象的 doc 实例
        initialize(doc);
    }
    /**
     *  AdvertisementTutorial，InputStream 对象作为参数的构造方法，抛出 IOE-
     xception
     */
    public AdvertisementTutorial(InputStream stream) throws IOException {
         //通过 StructuredDocumentFactory 的 newStructuredDocument()方法，构造
         一个名为 doc 的 StructuredTextDocument 对象实例
        StructuredTextDocument doc = (StructuredTextDocument)
                       StructuredDocumentFactory.newStructuredDocument(
                                   MimeMediaType.XMLUTF8, stream);
        //调用 initialize()方法，初始化 StructuredTextDocument 对象的 doc 实例
        initialize(doc);
    }
```

以上就是 AdvertisementTutorial 类的声明、构造方法的定义过程，它有 3 构造方法，根据参数的不同，构造不同的 AdvertisementTutorial 对象。

上文说过，AdvertisementTutorial 继承自 Advertisement 类，要完成 JXTA 中关于广告的相关功能，就需要重写父类所定义的相关方法。vertisementTutorial 还实现了 Comparable 接口，那么同样需要实现 Comparable 接口定义的方法，以下就是这些方法的实现。

```
/**
 * getDocument()方法，重写父亲 Advertisement 的方法，传入一个 MimeMediaType
 * 为参数，返回一个 Document 对象
 */
public Document getDocument(MimeMediaType asMimeType) {
    //通过 StructuredDocumentFactory 工厂类得到 StructuredDocument 对象
```

```
    StructuredDocument adv = StructuredDocumentFactory.
                newStructuredDocument(asMimeType, getAdvertisement
                Type());
    //判断 StructuredDocument 的实例所属的类型
    if (adv instanceof Attributable) {
         //类型转换，并在 adv 实例中添加属性信息
        ((Attributable) adv).addAttribute("xmlns:jxta", "http://jxta.
       org");
    }
    //初始化名为 e 的 Element 对象
    Element e;
    //通过 StructuredDocument 对象的 createElement()方法，根据主机信息的各种
    参数对 e 进行不同的赋值
    e = adv.createElement(idTag, getID().toString());
    //调用 StructuredDocument 的 appendChild()方法，加入实例 e
    adv.appendChild(e);
    //根据类名称的标签创建一个 Element
    e = adv.createElement(nameTag, getName().trim());
    adv.appendChild(e);
    //根据操作系统名称的标签创建一个 Element
    e = adv.createElement(OSNameTag, getOSName().trim());
    adv.appendChild(e);
    //根据操作系统版本信息的标签创建一个 Element
    e = adv.createElement(OSVersionTag, getOSVersion().trim());
    adv.appendChild(e);
    //根据操作系统架构信息的标签创建一个 Element
    e = adv.createElement(OSarchTag, getOSArch().trim());
    adv.appendChild(e);
    //根据 IP 地址信息的标签创建一个 Element
    e = adv.createElement(ipTag, getIP().trim());
    adv.appendChild(e);
    //根据硬件架构信息标签创建一个 Element
    e = adv.createElement(hwarchTag, getHWArch().trim());
    adv.appendChild(e);
    //根据提供商名称的标签创建一个 Element
    e = adv.createElement(hwvendorTag, getHWVendor().trim());
    adv.appendChild(e);
    //根据清单信息的标签创建一个 Element
    e = adv.createElement(swTag, getSWInventory().trim());
    adv.appendChild(e);
    //将所有赋值后的 adv 实例返回
    return adv;
}
/**
 * 重写父类的 getIndexFields()方法，直接返回一个表示广告索引的字符串数组
 */
public final String[] getIndexFields() {
    return fields;
}
/**
 * 重写父类的 getAdvertisementType()方法，返回一个标识消息类型的字符串
 */
public static String getAdvertisementType() {
    return "jxta:AdvertisementTutorial";
}
/**
 * 重写 Object 类的 equals()方法，自定义 equals 的条件
 */
```

```
public boolean equals(Object obj) {
    //对传入的 Object 对象参数进行判断
    if (this == obj) {
        return true;
    }
    //判断传入的对象类型是否是 AdvertisementTutorial 类型
    if (obj instanceof AdvertisementTutorial) {
        //如果是，则强制类型转换
        AdvertisementTutorial adv = (AdvertisementTutorial) obj;
        //通过 ID 是否 equals 来判断两个对象是否是 equals
        return getID().equals(adv.getID());
    }
    //否则返回 false
    return false;
}
/**
 * 实现 Comparable 接口的方法，Comparable 接口中只有一个 compareTo 方法，
 * AdvertisementTutorial 类实现了 Comparable 接口，所以必须要实现此方法。
 */
public int compareTo(Object other) {
    //通过 ID 对象的 toString 结果与传入对象的 toString 结果进行比较
    return getID().toString().compareTo(other.toString());
}
```

以上的几个方法分别与继承父类和实现接口有关，是完成广告消息处理的重要辅助方法。本类中，处理文档中的元素信息由 handleElement()方法来完成，此方法如下：

```
/**
 * handleElement()方法，用来从文档中处理单个元素，对每一个标签值进行 equals
 * 判断如果元素经过验证就返回 true;否则返回 False
 */
protected boolean handleElement(TextElement elem) {
    //对 idTag 进行判断
    if (elem.getName().equals(idTag)) {
        try {
            //通过 getTextValue 取出对应的值，赋给 URI 的 ID 对象
            URI id = new URI(elem.getTextValue());
            //再通过 setID()方法，设置此 URI 的 ID
            setID(IDFactory.fromURI(id));
        } catch (URISyntaxException badID) {
            //处理 URISyntaxException 异常
            throw new IllegalArgumentException("advertisement 中的未知 ID
            格式： " + elem.getTextValue());
        } catch (ClassCastException badID) {
            //处理 ClassCastException 异常
            throw new IllegalArgumentException("此 ID 不是已知的 ID 类型： " +
            elem.getTextValue());
        }
        return true;
    }
    //对 nameTag 进行判断
    if (elem.getName().equals(nameTag)) {
        setName(elem.getTextValue());
        return true;
    }
    //对 OSNameTag 进行判断
    if (elem.getName().equals(OSNameTag)) {
```

```
            setOSName(elem.getTextValue());
            return true;
        }
        //对 OSVersionTag 进行判断
        if (elem.getName().equals(OSVersionTag)) {
            setOSVersion(elem.getTextValue());
            return true;
        }
        //对 OSarchTag 进行判断
        if (elem.getName().equals(OSarchTag)) {
            setOSArch(elem.getTextValue());
            return true;
        }
        //对 ipTag 进行判断
        if (elem.getName().equals(ipTag)) {
            setIP(elem.getTextValue());
            return true;
        }
        //对 hwarchTag 进行判断
        if (elem.getName().equals(hwarchTag)) {
            setHWArch(elem.getTextValue());
            return true;
        }
        //对 hwvendorTag 进行判断
        if (elem.getName().equals(hwvendorTag)) {
            setHWVendor(elem.getTextValue());
            return true;
        }
        //对 swTag 进行判断
        if (elem.getName().equals(swTag)) {
            setSWInventory(elem.getTextValue());
            return true;
        }
        // 如果以上条件都不满足，则元素信息无法处理，返回 false
        return false;
    }
```

AdvertisementTutorial 类的作用是将主机系统的信息广播出去，在这个过程中，需要对消息按照 JXTA 规范进行定义，再对消息进行封装，最后对消息进行处理。在这一过程中，除了以上所说的方法外，还需要一个初始化的过程，这一过程由名为 initialize 的初始化方法完成，此方法的实现如下：

```
/**
 * initialize()方法，从构造文档的一部初始化系统广告，传入一个 Element 类型的对
 * 象实例
 */
protected void initialize(Element root) {
    //对传入的 Element 对象类型进行判断，是否为 TextElement 类型
    if (!TextElement.class.isInstance(root)) {
        //如果不是，则抛出异常
        throw new IllegalArgumentException(getClass().getName() + " 只支持
        TextElement 类型");
    }
    //进行强制类型转换，将 root 转换成 TextElement 类型
    TextElement doc = (TextElement) root;
    //对广告类型进行判断，是否与 doc 实例的名称满足 equals 条件
    if (!doc.getName().equals(getAdvertisementType())) {
```

```
        //如果不满足，则抛出异常
        throw new IllegalArgumentException(
                "Could not construct : " + getClass().getName() + "doc
                中包含一个" + doc.getName());
    }
    //从 doc 中通过 getChildren()方法，取出所有子结点，并赋给 Enumeration 对象
    Enumeration elements = doc.getChildren();
    //对 Enumeration 中的每一个元素进行遍历
    while (elements.hasMoreElements()) {
        //对第一个元素转换成 TextElement 类型
        TextElement elem = (TextElement) elements.nextElement();
        //如果无法处理 elem 实例，则抛出异常
        if (!handleElement(elem)) {
            LOG.warning("无法处理的 element \'" + elem.getName() + "\' in
            " + doc.getName());
        }
    }
}
```

在上述的几个方法中，用到了很多 setXX 和 getXX 形式的方法。这些方法是 AdvertisementTutorial 类中的用于信息存取的操作，这些方法的结构和形式都非常简单，请读者自己参考源代码，这里就不再说明了。

看过源代码的读者还会发现，在 AdvertisementTutorial 类还定义了一个名为 Instantiator 的静态内部类，它实现了 Instantiator 接口，用来完成与广告消息初始有关的操作。程序中，通过广告工厂（AdvertisementFactory）调用 registerAdvertisementInstance()方法来注册一个广告实例时，需要用到 Instantiator 对象，以下就是 Instantiator 静态内部类的具体实现。

注意： 本程序中只所以将 Instantiator 类定义一个静态内部类，是因为通常一个普通类是不允许声明为静态的，只有一个内部类才可以。这时这个声明为静态的内部类可以直接作为一个普通类来使用，而不需要实例化一个外部类，这样使用起来就很方便。

```
//静态内部类 Instantiator，实现了 Instantiator 接口
public static class Instantiator implements AdvertisementFactory.
Instantiator {
    /**
     * 返回一个通告的识别类型
     * @return String 返回一个 String 类型的通告信息
     */
    public String getAdvertisementType() {
        return AdvertisementTutorial.getAdvertisementType();
    }
    //返回一个 Advertisement 实例
    public Advertisement newInstance() {
        return new AdvertisementTutorial();
    }
    /**
     *  构造一个 Advertisement 实例，用来匹配 advertisementType 的类型
     */
    public Advertisement newInstance(net.jxta.document.Element root) {
        return new AdvertisementTutorial(root);
    }
}
```

注意： 在定义 Instantiator 类的时候，其实现的接口是 AdvertisementFactory.Instantiator。这种写法不同于其他的实现接口的写法，是因为 Instantiator 接口是在 AdvertisementFactory 类的内部定义的，所以在实现这个接口的时候需要外部的 AdvertisementFactory 类进行引导调用。

以上就是 AdvertisementTutorial 类的全部实现过程，至于测试广告效果的 main()方法读者可自己编写。根据本书提供的源代码，运行 main()方法后，会显示如图 14.6 所示的结果。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE jxta:AdvertisementTutorial>
<jxta:AdvertisementTutorial xml:space="default" xmlns:jxta="http://jxta.org">
        <ID>
                urn:jxta:jxta-Null
        </ID>
        <name>
                AdvertisementTutorial
        </name>
        <OSName>
                Windows XP
        </OSName>
        <OSVer>
                5.1
        </OSVer>
        <osarch>
                x86
        </osarch>
        <ip>
                192.168.1.77
        </ip>
        <hwarch>
                x86
        </hwarch>
        <hwvendor>
                Sun Microsystems Inc.
        </hwvendor>
        <sw/>
</jxta:AdvertisementTutorial>
```

图 14.6　JXTA 通告实例程序运行后的结果

由图 14.6 所示的结构可以清楚地看出，这是一个用 XML 文件描述的通告信息，通告的内容表示的就是系统有关的信息，包括 ID、类名称、操作系统名称、操作系统版本、IP 地址等信息。由此说明，通过 AdvertisementTutorial 类可以得到主机的消息，并将这些消息封装成 XML 广告出来，这与类的设计目标是一致的。

在以上的整个实现过程中，总结一下就是，在 JXTA 平台中创建一个通告的时候，主要是应用这句代码：

```
AdvertisementFactory.registerAdvertisementInstance(
AdvertisementTutorial.getAdvertisementType(),new
AdvertisementTutorial.Instantiator());
```

以上代码就是通过工厂通告来注册一个通告的实例过程。

14.5.3　JXTA 的消息

JXTA 的消息是对等体之间数据交换的基本单元。这些消息通过管道在不同的服务之间发送或接收，由服务所实现的 JXTA 协议都会发送或接收 JXTA 消息。JXTA 使用了二进制格式的消息保证二进制或者 XML 数据的高效传输。每个 JXTA 传输绑定都会采用大家公认最合适的方式。例如：JXTA 的参考中所使用的 TCP/IP 和 HTTP 传输绑定都使用二进制的消息格式。

消息是一组具有名称和具有类型的内容，这些内容被称为元素。因此，消息本质上就是一些名称/数据值对的集合。这些内容可以是任何类型。许多核心服务把 XML 通告作为消息元素内容来发送。当一条消息传入一个协议栈时（应用程序、服务及传输），每一级都会向该消息添加一个或多个具名的元素。当消息返回到堆栈的顶部时，该协议会清除这些元素，消息就是一系列有序的元素，最后添加上的元素将出现在该消息的末端。

下面就用一个实例程序来演示一下，在 JXTA 平台中应用 JXTA 消息的实现过程。本实例程序命名为 MessageTutorial.java，此类位于 jxta_example 工程中的 p2p.jxta.tutorial 包下。以下是 MessageTutorial 类中主要方法的说明，详细完整的实现源代码请读者参考随书光盘，MessageTutorial 类的文件位置如下：

【源代码：ch14\ch14_code\jxta_example\src\p2p\jxta\tutorial\MessageTutorial.java】

MessageTutorial 类是一个简单的操作 JXTA 消息的例子，程序通过消息元素分别来添加和读取 String 型、Int 型、long 型、对象型、byte 数组型等数据信息。

下面先讲解一下增加不同的数据类型到 JXTA 消息中的实现方法，代码如下：

```
//定义一个 private 的静态的 MimeMediaType 类型的常量
private final static MimeMediaType GZIP MEDIA TYPE = new MimeMediaType
("application/gzip").intern();
/**
 *addStringToMessage()方法，添加一个 String 型数据到 JXTA 消息，形成一个
StringMessageElement
 *
 * @param message   需要加入的消息
 * @param nameSpace 需要加入的元素的名称空间.值为 NUll 的时候，指的是默认的名称空间。
 * @param elemName  元素的名称。
 * @param data      需要加入到 LongToMessage 属性的特征
 */
public static void addStringToMessage(Message message, String nameSpace,
String elemName, String string) {
    //调用 message 对象的 addMessageElement()方法，传入名称空间和 StringMessage-
    Element 对象作为参数
    message.addMessageElement(nameSpace, new StringMessageElement
   (elemName, string, null));
}
/**
 * 添加一个 long 型数据到消息
 *
 * @param message   需要加入的消息
 * @param nameSpace 需要加入的元素名称空间。值为 NUll 的时候，指的是默认的名称空间。
 * @param elemName  元素的名称。
 * @param data      需要加入到 LongToMessage 属性的特征
 */
public static void addLongToMessage(Message message, String nameSpace,
String elemName, long data) {
    //调用 message 对象的 addMessageElement()方法，传入名称空间和 StringMess-
    ageElement 对象作为参数
    message.addMessageElement(nameSpace, new StringMessageElement(elem-
    Name, Long.toString(data), null));
}
/**
```

```
     * 添加一个 Int 类型数据到消息
 * @param message   需要加入的消息
 * @param nameSpace 需要加入的元素名称空间。值为 NUll 的时候，指的是默认的名称空间。
 * @param elemName  元素的名称。
 * @param data      需要加入到 LongToMessage 属性的特征
 */
public static void addIntegerToMessage(Message message, String nameSpace,
String elemName, int data) {
     //调用 message 对象的 addMessageElement()方法，传入名称空间和 StringMessage
     Element 对象作为参数
    message.addMessageElement(nameSpace, new StringMessageElement
   (elemName, Integer.toString(data), null));
}
/**
 * 添加一个 byte 组数据到消息
 * @param message   需要加入的消息
 * @param nameSpace 需要加入的元素的名称空间.值为 NUll 的时候，指的是默认的名称空间.
 * @param elemName  元素名称
 * @param data      字节数据
 * @param compress  是否需要用 GZIP 进行压缩
 * @throws IOException  如果读写发生错误时将抛出 IOException
 */
public static void addByteArrayToMessage(Message message, String nameSpace,
String elemName, byte[] data, boolean compress) throws IOException {
    //初始化一个字节数组
     byte[] buffer = data;
     //得到一个 MimeMediaType 对象实现
    MimeMediaType mimeType = MimeMediaType.AOS;
    //判断是否需要用 GZIP 进行压缩
    if (compress) {
         //定义一个字节数组输出流对象
        ByteArrayOutputStream outStream = new ByteArrayOutputStream();
        //通过输出流构造一个 GZIPOutputStream 对象
        GZIPOutputStream gos = new GZIPOutputStream(outStream);
        //写入数据
        gos.write(data, 0, data.length);
        gos.finish();
      //关闭 GZIPOutputStream 输出流
        gos.close();
         //将 ByteArrayOutputStream 输出流转换为字节数组存入到 buffer 中
        buffer = outStream.toByteArray();
         //将 mimeType 赋值为定义的 GZIP_MEDIA_TYPE 类型
        mimeType = GZIP_MEDIA_TYPE;
    }
    message.addMessageElement(nameSpace, new ByteArrayMessageElement(elem
    Name, mimeType, buffer, null));
}
/**
 * 通过指定的名称空间和指定的元素名称来添加一个对象型的数据到消息
 * @param message   需要添加对象的消息
 * @param nameSpace 名称空间
 * @param elemName  给定的元素名称
```

```
 * @param object    对象实例
    * @throws IOException if an io error occurs
    */
public static void addObjectToMessage(Message message, String nameSpace,
String elemName, Object object) throws IOException {
    //定义一个字节数组输出流对象
    ByteArrayOutputStream bos = new ByteArrayOutputStream();
    //通过字节数组输出流对象 bos 构造一个 ObjectOutputStream 对象
    ObjectOutputStream oos = new ObjectOutputStream(bos);
    //调用 ObjectOutputStream 的 write()方法写入 Object 实例
    oos.writeObject(object);
    oos.close();
    bos.close();
    //添加字节数组到消息
    addByteArrayToMessage(message, nameSpace, elemName, bos.toByteArray(),
    false);
}
```

有了以上的将不同数据类型加入到消息中的方法，就有与之对应的从消息中取出相应数据类型的方法，这些方法的具体实现如下：

```
/**
 * getStringFromMessage()方法，从 JXTA 消息里取出一个字符串（String）数据
 */
public static String getStringFromMessage(Message message, String
nameSpace, String elemName) {
    //通过 message 的 getMessageElement()方法，得到一个 MessageElement 对象
    MessageElement me = message.getMessageElement(nameSpace, elemName);
    //对 MessageElement 对象进行判断
    if (null != me) {
        //如果不为空，则直接返回 MessageElement 对象 toString 方法的结果
        return me.toString();
    } else {
        //否则返回 null
        return null;
    }
}
/**
 * getLongFromMessage()方法，从 JXTA 消息里取出一个 Long 型数据
 */
public static long getLongFromMessage(Message message, String nameSpace,
String elemName) throws NumberFormatException {
    //调用 getStringFromMessage()方法，取得一个用字符串表示的 Long 型数据
    String longStr = getStringFromMessage(message, nameSpace, elemName);
    //判断字符串值是否为 null
    if (null != longStr) {
        //如果不为 null，直接将字符串转换成 long 型数据即可
        return Long.parseLong(longStr);
    } else {
        //否则抛出异常
        throw new NumberFormatException("没有此消息元素.");
    }
}
/**
 * getIntegerFromMessage()方法，从 JXTA 消息里取出一个 Int 型数据
 */
public static int getIntegerFromMessage(Message message, String nameSpace,
```

```
String elemName) throws NumberFormatException {
    //调用 getStringFromMessage()方法，取得一个用字符串表示的 int 型数据
    String intStr = getStringFromMessage(message, nameSpace, elemName);
    //判断字符串值是否为 null
    if (null != intStr) {
        //如果不为 null，直接将字符串转换成 int 型数据即可
        return Integer.parseInt(intStr);
    } else {
       //否则抛出异常
        throw new NumberFormatException("没有此消息元素.");
    }
}
/**
 * getInputStreamFromMessage()方法，从 JXTA 消息取出字节数据输入流对象
 */
public static InputStream getInputStreamFromMessage(Message message,
String nameSpace, String elemName) throws IOException {
     //定义一个字节输入流
    InputStream result = null;
    //通过 message 的 getMessageElement()方法，得到一个 MessageElement 对象
    MessageElement element = message.getMessageElement(nameSpace,
    elemName);
    //对 MessageElement 对象进行判断
    if (null == element) {
         //如果为空则程序直接返回
        return null;
    }
    //对 element 的数据类型进行判断，是否是 GZIP_MEDIA_TYPE 形式的压缩类型
    if (element.getMimeType().equals(GZIP_MEDIA_TYPE)) {
         //如果则通过 GZIPInputStream 对象读取数据
        result = new GZIPInputStream(element.getStream());
     //如果是 MimeMediaType.AOS 类型的数据
    } else if (element.getMimeType().equals(MimeMediaType.AOS)) {
         //直接调用 getStream()方法读取
        result = element.getStream();
    }
     //返回读取结果
    return result;
}
/**
 * getObjectFromMessage()方法，从 JXTA 消息里读取一个单一的 Java 对象
 */
public static Object getObjectFromMessage(Message message, String nameSpace,
String elemName) throws IOException, ClassNotFoundException {
    //通过 getInputStreamFromMessage()方法，取得一个字节输入流对象
     InputStream is = getInputStreamFromMessage(message, nameSpace,
     elemName);
    //对输入流对象进行判断，是否为 null
    if (null == is) {
        return null;
    }
     //不为 null 的时候，将 InputStream 实例作为参数，封装一个 ObjectInputStream
     对象
    ObjectInputStream ois = new ObjectInputStream(is);
    //直接调用 ObjectInputStream 对象的 readObject()方法，读取数据并返回
    return ois.readObject();
}
```

以上的方法，既完成了在消息中添加数据，也可以从消息中读取数据。下面就用几个方法来具体地验证在消息中添加和读取数据的过程，具体实现如下：

```
//处理字符串类型数据的例子
public static void stringExample() {
     //初始化 Message 对象
    Message message = new Message();
     //调用 addStringToMessage()方法，在消息中加入字符串类型的数据
    addStringToMessage(message, "TutorialNameSpace", "String Test", "This
    is a test");
     //调用 printMessageStats()方法，打印消息
    printMessageStats(message, true);
     //打印输出字符串值
    System.out.println("String 值:" + getStringFromMessage(message, "Tutor-
    ialNameSpace", "String Test"));
}
//处理 Long 类型数据的例子
public static void longExample() {
    //初始化 Message 对象
    Message message = new Message();
    //调用 addLongToMessage()方法，在消息中加入 Long 类型的数据
    addLongToMessage(message, "TutorialNameSpace", "long test", Long.
    MAX_VALUE);
    //调用 printMessageStats()方法，打印消息
    printMessageStats(message, true);
    //打印输出 Long 型数据值
    System.out.println("long 值 :" + getLongFromMessage(message, "Tutor-
    ialNameSpace", "long test"));
}
//处理 int 类型的数据的例子
public static void intExample() {
     //初始化 Message 对象
    Message message = new Message();
    //调用 addIntegerToMessage()方法，在消息中加入 int 类型的数据
    addIntegerToMessage(message, "TutorialNameSpace", "int test", Integer.
    MAX_VALUE);
    //调用 printMessageStats()方法，打印消息
    printMessageStats(message, true);
    //打印输出 int 型数据值
    System.out.println("int 值 :" + getIntegerFromMessage(message, "Tutori-
    alNameSpace", "int test"));
}
//处理字节数组类型数据的例子
public static void byteArrayExample() {
    //初始化 Message 对象
    Message message = new Message();
    try {
         //初始化一个名为 message.tst 的文件
        File file = new File("message.tst");
         //调用 File 对象的 createNewFile()方法，创建此文件
        file.createNewFile();
     //通过 message.tst 文件作为参数，封装一个 RandomAccessFile 对象，此文件属性可
     读可写
        RandomAccessFile raf = new RandomAccessFile(file, "rw");
        //设置文件长度
        raf.setLength(1024 * 4);
```

```
        //定义大小
        int size = 4096;
        //根据指定大小初始化一个字节数组
        byte[] buf = new byte[size];
        //调用 RandomAccessFile 的 read()方法，读取文件内容
        raf.read(buf, 0, size);
        //通过 addByteArrayToMessage()方法，传入 buf 参数，添加字节数组数据到消息
        addByteArrayToMessage(message, "TutorialNameSpace", "byte test",
        buf, true);
        //打印消息状态
        printMessageStats(message, true);
        //通过 getInputStreamFromMessage()方法，取得 InputStream 字节输
        入流
        InputStream is = getInputStreamFromMessage(message, "Tutorial-
        NameSpace", "byte test");
        int count = 0;
        //读取此数据流直接结
        while (is.read() != -1) {
            count++;
        }
    } catch (IOException io) {
        io.printStackTrace();
    }
}
/**
 * printMessageStats 用来打印消息的状态信息
 */
public static void printMessageStats(Message msg, boolean verbose) {
    try {
        System.out.println("------------消息开始-------------");
        //通过 WireFormatMessageFactory 工厂类的 toWire()方法，写入消息
        WireFormatMessage serialed = WireFormatMessageFactory.toWire(
                            msg, new MimeMediaType("application/x-jxta-
                            msg"), null);
        //打印消息大小
        System.out.println("Message 大小为 :" + serialed.getByteLeng-
        th());
        //取得 ElementIterator 实例
        ElementIterator it = msg.getMessageElements();
        //遍历 ElementIterator
        while (it.hasNext()) {
            //取出所有的 MessageElement
            MessageElement el = it.next();
            //将得到的结果打印输出
            System.out.println("Element : " + it.getNamespace() + " :: "
                                           + el.getElementName());
            //根据 verbose 参数，是否打印 MessageElement
            if (verbose) {
                System.out.println("[" + el + "]");
            }
        }
        System.out.println("------------消息结束---------------");
    //捕获并处理异常
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

以上的方法，包含了 JXTA 消息中各种数据类型的添加与读取，只要在 main()方法中对这些方法进行简单地调用，就可以查看它们的执行结果了。

在 JXTA 中，所有的消息基本是 String、Long、Int、ByteArray、Object 等几种类型组成，有了以上的对这几种数据类型的处理，就可以在 JXTA 的消息中添加任意消息，也可以从接收到的 JXTA 消息中，读取任何类型的数据。

关于 JXTA 的消息就讲到这里，读者在学习此知识点的时候，要结合 MessageTutorial 类的源代码，理解各种类型的数据是如何加入到消息中的，又是如何从消息中读取的，最好能根据实例实际编写代码并运行观察效果，这样会理解的更深入。

14.6　本章小结

JXTA 作为专门针对 P2P 的解决方案，它寻求通过提供一套所有 P2P 应用程序都能使用的标准来清除 P2P 技术发展中的障碍，从而形成一个通用的满足现代需要的分布式 P2P 计算平台。

JXTA 作为 P2P 的重要解决之道，要学习 P2P 技术、进行与 P2P 应用有关的开发，就有必要学习 JXTA 的相关知识。本章从最基础的层面讲解了 JXTA 的基本术语、协议、开发方法及一些概念性的知识，这些知识只是为读者提供一个简单的认识 JXTA 的视角，无法满足深入学习的需要。希望读者能在理解本章知识的基础上，不断实践、寻求新的学习素材和途径，真正地学好、用好 JXTA 技术。